GitHub repository: https://github.com/acse-2020/acse2020-acse9-finalreport-acse-tc20.git

Imperial College London
Department of Earth Science and Engineering

MSc in Applied Computational Science and Engineering
Independent Research Project
Final Report

Development of a Hybrid Power Plant Model and Sizing Tool as a Platform for Advanced Optimisation and Modelling Functionality

by
Teddy Cheung

teddy.cheung20@imperial.ac.uk

GitHub ID: acse-tc20

Supervisors:
Prof. Matthew Piggott

Industry Collaboration:
Dr. Roy Brown (Frazer-Nash)
Zoe Goss (Frazer-Nash)

GitHub repository: https://github.com/acse-2020/acse2020-acse9-finalreport-acse-tc20.git

ABSTRACT

Key words: *Hybrid Power Plant, Sizing Optimization, HOMER*

A Hybrid Power Plant (HPP) seeks to improve the predictability, dispatchability, and economics of stand-alone intermittent generation by combining different types of generation and storage assets. Optimal HPP sizing can be formulated as a multi-objective, multi-variate optimisation problem. The industry standard software for HPP sizing optimisation is HOMER Pro. HOMER Pro allows the user to optimise the capacity each of desired generation/storage asset, with the objective to minimise Net Present Cost (NPC) subject to several optional constraints, such as the ability of the HPP to fulfil a representative load profile. Because HOMER Pro is proprietary and utilises computationally inefficient exhaustive search, there exists an opportunity for an open-source software to be developed which enables higher complexity HPP models with state-of-the-art global optimisation methods. This project developed an open-source HPP sizing software (the ACSE9 tool), which uses power output and financial models verified and validated by HOMER Pro results. Validation was performed for a range of geographical scenarios; the equator test case achieved a 0.031%, 0.161% and 6.35% error against HOMER Pro's wind, solar and storage power output models respectively. The ACSE9 tool was developed with the ability to optimise using exhaustive search, and with pre-packaged modules such as `scipy.optimize`. Using SciPy's differential evolution optimiser, the ACSE9 tool found a solution which was 25% cheaper NPC than discrete exhaustive search with a resolution of 10 components. In addition to its optimisation options, the ACSE9 tool provides the ability to script HPP simulations and sensitivity analyses, which the industry standard does not.

PROJECT OBJECTIVES

1. Develop an open-source HPP sizing tool with a theoretical basis comparable to that offered by HOMER PRO.
2. Develop and give the user an option to use multiple global optimisation methods for the sizing of HPPs.
3. Validate the open-source model and optimisation methods by comparison to HOMER Pro solutions.
4. In general, provide a software architecture which allows for integration of high accuracy models and pre-packaged global optimisation functions; such that the tool can provide a higher-fidelity, scriptable alternative to the industry standard.

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Professor Matthew Piggott, for the opportunity to complete this project, and for invaluable guidance on the scope and directions of its research. I would also like to thank my collaborators from Frazer-Nash, Dr. Roy Brown and Zoe Goss, for their more than generous time commitments and insight they gave throughout the project. I would like to extend my thanks to Dr. James Percival and the Katara Study Group for helping me to maintain progress and avoid any obstacles to the flow of research. Not least, I would like to thank my friends and family, who have supported me through my education thus far.

DICTIONARY

3.14 docs	Documentation for v3.14 of HOMER PRO: <https://www.homerenergy.com/products/pro/docs/3.14/index.html>
Author	Teddy Cheung
CapEx	Capital Expenditure
CSF	Capacity Shortage Fraction
Eq	Equator
GA	Genetic Algorithm
GHG	Green House Gas
HOMER	Hybrid Optimization of Multiple Energy Resources
HPP	Hybrid Power Plant
NH	Northern Hemisphere
NOCT	Nominal Cell Operating Temperature
NPC	Net Present Cost
OpEx	Operational Expenditure
PSO	Particle Swam Optimisation
PV	Photovoltaic
SD	Standard Air Density
SH	Southern Hemisphere
SoC	State-of-Charge
STC	Standard Test Conditions
STP	Standard Temperature Pressure
WTG	Wind Turbine Generator

1. INTRODUCTION

The ‘energy transition’ away from fossil fuel-based generation toward low-carbon sources has increased the presence of intermittent electricity generation, such as wind turbines and solar photovoltaics, into both on-grid and off-grid power systems (World Bank, 2021). The energy transition is driven by a mixture of personal incentives and predefined quotas between policy makers, power system operators, project developers, and lenders. There is consensus that an optimal electricity generation source would be one which produces few, or preferably no, greenhouse gases over its lifecycle, has competitive capital and operational costs, and provides predictable or dispatchable (with little ramp-up time) generation (Climate Change Committee, 2020).

Globally, modern onshore wind and utility scale solar photovoltaics (solar PV) are cheaper than or cost competitive with traditional fossil fuel generation, such as coal fire plants and combined cycle gas turbines (Schlömer, et al., 2014). The generation of these intermittent sources is predictable, to a reasonable level, at a long-term scale through probabilistic energy yield assessments, and a medium-term scale through weather forecasting. However, their stand-alone generation is not dispatchable. Also, although generation is predictable, it will not necessarily be matched with the load required for the power system. When the power generated by such intermittent sources exceeds the required load the generation must be curtailed. If the generation is in deficit to the required load, blackouts can occur. Thus, the correct sizing of an intermittent generation asset is an important issue.

A Hybrid Power Plant (HPP) seeks to improve the predictability, dispatchability, and economics of stand-alone intermittent generation by combining different types of generation and storage assets. For example, in a region where irradiance and wind speed are negatively correlated (e.g., Monforti, et al., 2013), onshore wind and solar PV generation could be developed on the same site to smooth the generation profile (Paska, et al., 2009). Additionally, storage assets such as grid-scale Li-ion batteries could be included on the site to store surplus energy which would be curtailed otherwise.

The design of a HPP requires choices to be made about the technologies used and the relative sizing of each component, with the major constraint of matching a desired load profile. Therefore, this design can be presented as a multi-objective, multi-variate optimisation problem. A common cost function variable to optimise is the Net Present Cost (NPC) of the HPP, which is the cost to acquire, install, operate and maintain the given HPP design over its lifetime (HOMER Energy, 2020). Additional constraint variables can be included in the optimisation, including lifecycle Green House Gas (GHG) emissions, the expected plant capacity factor, and plant footprint, among a variety of other costs.

The Hybrid Optimization of Multiple Energy Resources (HOMER) software (HOMER Energy, 2021) is the most widely used tool for HPP sizing optimisation (Anoune, et al., 2018). HOMER uses exhaustive search[1] to search and rank possible solutions by NPC to produce an ‘optimal solution’.

In sizing optimisation, the search space contains all the combinations of the different capacities of each generation/ storage assets. For instance, a search space of 10 to 20 Solar PV modules and 90 to 100 Li-ion storage cells, with a resolution of 10 components, would require the evaluation of 10+90, 10+100, 20+90 and 20+100 modules and cells, respectively. Since a user’s search-space can be as large as desired, any tool which uses exhaustive search must make several assumptions in its modelling decisions to lower computational cost and provide a reasonable runtime. However, it is possible to use other optimisation methods for the sizing of a HPP. Anoune, et al., (2018) provide a detailed review of optimisation methods which have been used for the sizing of PV-wind HPPs. This review includes a summary of the key software packages used in industry and a review and comparison of optimisation methods used by researchers. Some state-of-the-art optimisation techniques discussed include Particle Swarm Optimisation (PSO), Genetic Algorithm (GA) and Multi-Objective Approaches.

Regardless of the optimisation method used, it is likely that a model of the HPP will still be required for any sizing task. The modelling of a HPP requires extensive

1. HOMER also includes the option to use a proprietary ‘HOMER optimisation’ optimisation method instead of exhaustive search. Since there is little available information on the methodology used by ‘HOMER optimisation’, it is not discussed in this report.

physical and economic views of each component of the plant, as well as consideration of the system in which it operates. Commonly, the crucial items to be modelled are: A representative load profile; resource profiles for weather-based generation; power production of a given generation asset; battery state-of-charge; and the Capital Expenditure (CapEx) and Operational Expenditure (OpEx) costs of each generation/storage asset.

This report describes version 0.1 (beta) of an open-source HPP modelling and sizing optimisation tool – named the ACSE9 tool for the remainder of this report. The current version of the ACSE9 tool has six main functionalities: wind, solar PV, and Li-ion power output models; a discounted cash flow model; an exhaustive-search-based sizing optimisation template; and an evolutionary algorithm-based optimisation template. All functionalities (apart from the evolutionary algorithm-based optimisation) are based on and have been validated against HOMER and achieve validation to a reasonable level. Thus, the ACSE9 tool provides an open-source platform upon which advanced component modelling and optimisation algorithms can be built. In addition, the ACSE9 tool provides the ability to script HPP sizing simulations – an option which is not currently available in HOMER Pro. The key benefit to developing such an open-source platform is that more accurate optimal solutions may be found by enabling higher-accuracy models with advanced optimisation techniques. Further, the sensitivity of these new optimal solutions can be explored in more detail when given the opportunity to script analyses.

Section 2 of this report provides a description of the software design, including its architecture, modelling decisions, and validation and verification procedures. Section 3 presents results and a discussion of the optimal solutions provided by the ACSE9 tool and the associated runtimes. Section 4 summarises the work to date and proposes avenues for future work.

2. SOFTWARE DESCRIPTION

In this section, the architecture, routines, modelling decisions, and validation routines of the ACSE9 tool are described. The entirety of the ACSE9 tool was written in python (3.3.8), and its source code can be found in the GitHub Repository located at <https://github.com/acse-2020/acse2020-acse9-finalreport-acse-tc20.git>. A full documentation of the power output modelling literature can be accessed in Appendix 1.

a. ARCHITECTURE

The purpose of the ACSE9 tool is to produce an optimal combination of Wind Turbine Generators (WTGs), solar photovoltaic (solar PV) modules and Li-ion cells, to satisfy a pre-determined load profile. The optimal solution is so-called because it has the lowest NPC compared to other combinations which satisfy the load (i.e., 'feasible' HPP configurations). To perform this modelling-optimisation workflow, the ACSE9 tool requires six inputs: Representative irradiance and wind speed profiles for the plant location; a minimum load profile which must be satisfied for any HPP configuration to be considered feasible; and physical/financial specifications of the components to be used in the HPP design.

Figure 2.1 illustrates an overview of the code architecture for the ACSE9 tool. This architecture was developed by the author and was based on the calculations used by the HOMER Pro v3.14– the off-grid HPP sizing tool sold by HOMER Energy. The calculations used by HOMER Pro v.3.14 can be found at <https://www.homerenergy.com/products/pro/docs/3.14/index.html>, a full description of how these calculations are used in the ACSE9 tool can be found in Appendix 1.

The architecture illustrated in Figure 2.1 is an exhaustive-search-based workflow in which every possible combination of the individual WTG, solar PV and Li-ion cell search-spaces are evaluated. The feasibility of any one combination is based on its ability to achieve a maximum Capacity Shortage Fraction (CSF) which equals the ratio of the total annual capacity shortage to the total annual electrical demand. Note, all analyses carried out in this report used a value of CSF = 1%. As discussed in the introduction, exhaustive search is not only computationally inefficient, but can require simplifications in model design to reduce overall computational cost. Demonstrative of this fact is that exhaustive search had to be made discrete and the resolution increased in order to run simulations with an acceptable computational load during this project (see Section 3). Alternative optimisation techniques will be discussed in more detail in Section 2.e, yet, considering the code architecture, the majority of alternative techniques require the power output, feasibility, and financial model subroutines shown in Figure 2.1. The key difference of alternative optimisation techniques is the traversal of the search-space. For example, an evolutionary algorithm would interpret the search-space not as a set of configurations to evaluate, but as a set of boundary conditions within which configurations can be generated.

b. POWER OUTPUT ALGORITHMS

The ACSE9 tool contains models which simulate the power output at every timestep of an array of generation or storage components. \`ACSE9.WindModel\` models an array of WTGs, \`ACSE9.SolarModel\` models an array of solar PV modules, and \`ACSE9.StorageModel\` models a collection of Li-ion cells. All the power output models presented in this report, and available in the ACSE9 tool (as of v.0.1), are based on the calculations used by HOMER Pro v3.14 – the off-grid HPP sizing tool sold by HOMER Energy. These calculations can be found at <https://www.homerenergy.com/products/pro/docs/3.14/index.html>: Any reference made to the equations found within this documentation will be denoted as (3.14 docs) for the remainder of this report.

The power output models are defined as python classes, namely \`ACSE9.WindModel.WindArray\`, \`ACSE9.SolarModel.SolarArray\`, and \`ACSE9.StorageModel.StorageArray\`. These definitions allow for efficient scripting where the class attributes and methods associated with each model can be varied for any desired sensitivity analyses.

As noted, a full description of the theory behind each model can be found in the documentation folder of the GitHub Repository (<https://github.com/acse-2020/acse2020-acse9-finalreport-acse-tc20.git>). Below, an overview of the code structure of the power output models is given, along with the class diagram for each model (Figure 2.2). Please note, the class diagrams are included in order to assist the reading of the algorithm pseudocode. For a full description of each class attribute and method, refer to Appendix 1.

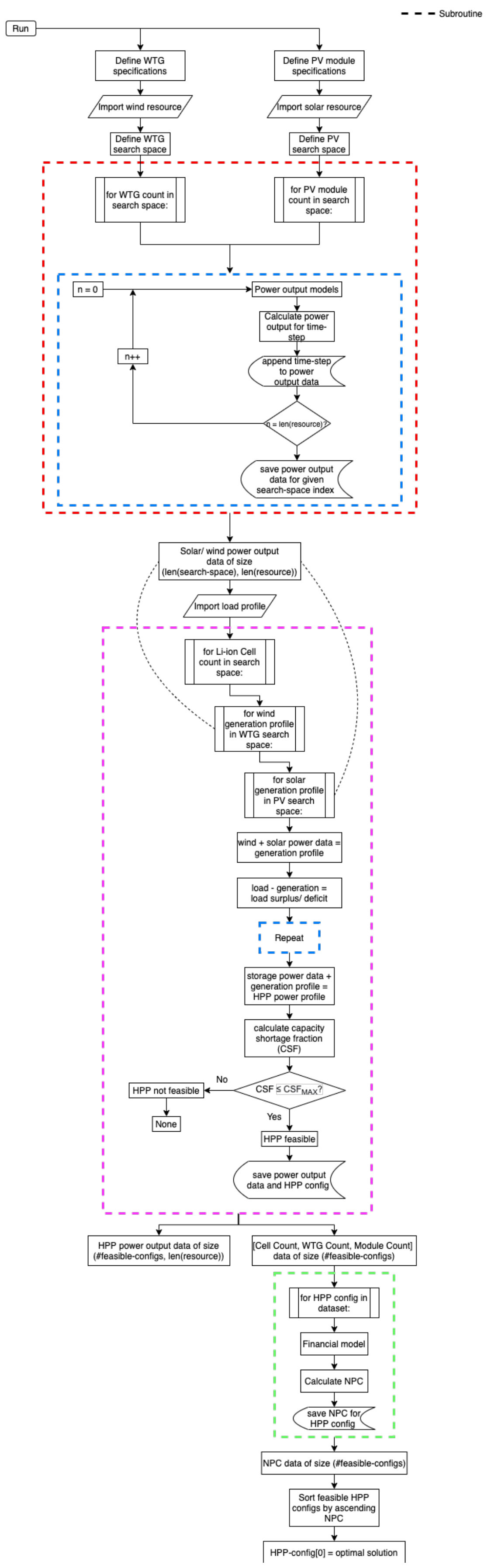

Subroutine
Run
Define WTG specifications
Define PV module specifications
Import wind resource
Import solar resource
Define WTG search space
Define PV search space
for WTG count in search space:
for PV module count in search space:
n = 0
Power output models
Calculate power output for time-step
n++
append time-step to power output data
n = len(resource)?
save power output data for given search-space index
Solar/ wind power output data of size (len(search-space), len(resource))
Import load profile
for Li-ion Cell count in search space:
for wind generation profile in WTG search space:
for solar generation profile in PV search space:
wind + solar power data = generation profile
load - generation = load surplus/ deficit
Repeat
storage power data + generation profile = HPP power profile
calculate capacity shortage fraction (CSF)
CSF ≤ CSF_MAX?
No
HPP not feasible
None
Yes
HPP feasible
save power output data and HPP config
HPP power output data of size (#feasible-configs, len(resource))
[Cell Count, WTG Count, Module Count] data of size (#feasible-configs)
for HPP config in dataset:
Financial model
Calculate NPC
save NPC for HPP config
NPC data of size (#feasible-configs)
Sort feasible HPP configs by ascending NPC
HPP-config[0] = optimal solution

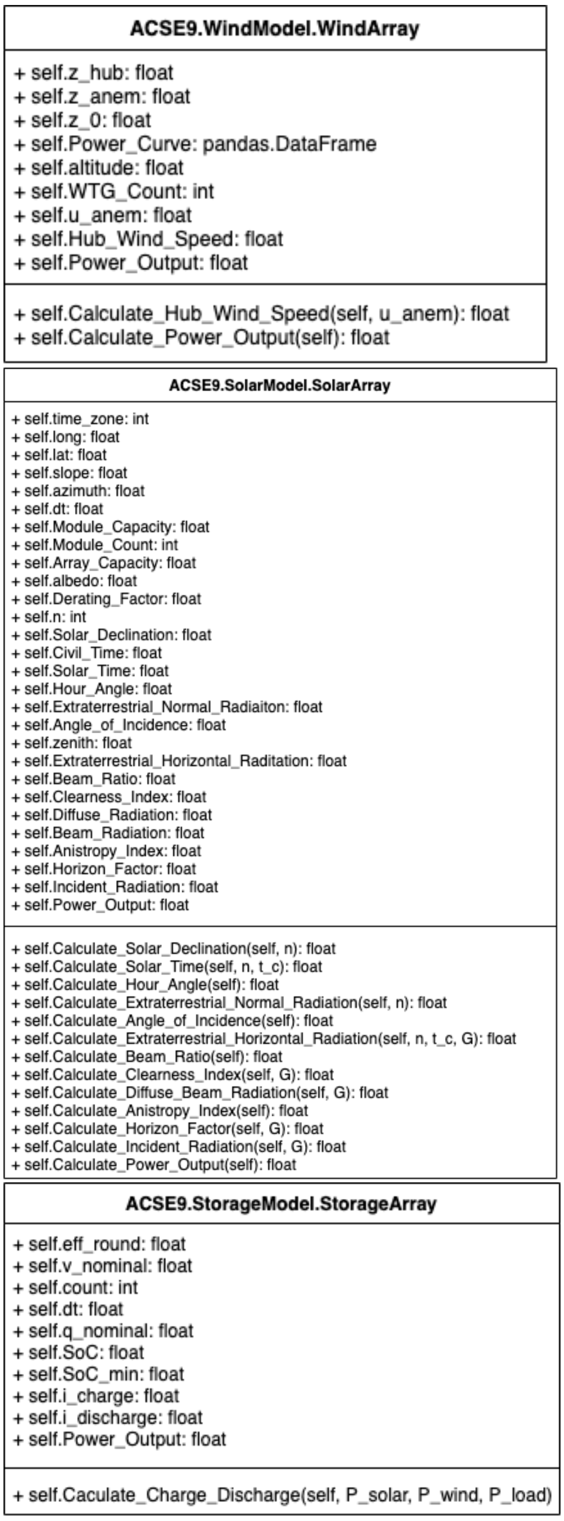
Figure 2.1: Code Architecture (Exhaustive Search)

Wind Model

With regards to number of operations and dependencies, the ACSE9 tool wind power output model is a relatively simple power-curve-based module. The model requires two inputs (additional to the WTG physical specifications): A resource profile of wind-speed (in m.s^{-1}) at anemometer height (in m), at every time-step; and a representative power curve for which the WTG is to be modelled. A power curve is a dataset which contains manufacturer data on how the power output (in kW) of a single WTG varies with the wind speed at hub-height (in m), at a Standard Air Density (SD), ρ_0, of 1.225 kg.m^{-3}. The calculation of the power output of the WTG array at each time step has three stages. First, the wind speed at WTG hub-height is calculated by adjusting the wind speed at anemometer height which is taken from the imported resource profile. After the wind speed at hub-height has been calculated, the expected SD power output of the WTG array at the current time-step is calculated performing linear interpolation of the imported power curve. Finally, the predicted WTG power output is calculated by adjusting the interpolation result for the air density at the WTG array location. Air density is assumed to be a function of altitude only.

ACSE9.WindModel Algorithm (time-step)

```
Inputs   hub height, anemometer height, surface
         roughness, altitude, manufacturer power
         curve data.
1:       Define array inside class:
         `ACSE9.WindModel.WindArray`
2:       data in
         wind speed at current time-step
3:       Calculate wind speed at hub height[2]
4:       if hub wind speed > power curve cut-off:
           self.Power_Output = 0.0
4:       else:
           Interpolate
           Power curve around hub wind speed[3]
5:         Calculate
           Power output adjusted for air density[4]:
           self.Power_Output
6:       end
```



```
ACSE9.WindModel.WindArray
+ self.z_hub: float
+ self.z_anem: float
+ self.z_0: float
+ self.Power_Curve: pandas.DataFrame
+ self.altitude: float
+ self.WTG_Count: int
+ self.u_anem: float
+ self.Hub_Wind_Speed: float
+ self.Power_Output: float

+ self.Calculate_Hub_Wind_Speed(self, u_anem): float
+ self.Calculate_Power_Output(self): float
```

```
ACSE9.SolarModel.SolarArray
+ self.time_zone: int
+ self.long: float
+ self.lat: float
+ self.slope: float
+ self.azimuth: float
+ self.dt: float
+ self.Module_Capacity: float
+ self.Module_Count: int
+ self.Array_Capacity: float
+ self.albedo: float
+ self.Derating_Factor: float
+ self.n: int
+ self.Solar_Declination: float
+ self.Civil_Time: float
+ self.Solar_Time: float
+ self.Hour_Angle: float
+ self.Extraterrestrial_Normal_Radiaiton: float
+ self.Angle_of_Incidence: float
+ self.zenith: float
+ self.Extraterrestrial_Horizontal_Raditation: float
+ self.Beam_Ratio: float
+ self.Clearness_Index: float
+ self.Diffuse_Radiation: float
+ self.Beam_Radiation: float
+ self.Anistropy_Index: float
+ self.Horizon_Factor: float
+ self.Incident_Radiation: float
+ self.Power_Output: float

+ self.Calculate_Solar_Declination(self, n): float
+ self.Calculate_Solar_Time(self, n, t_c): float
+ self.Calculate_Hour_Angle(self): float
+ self.Calculate_Extraterrestrial_Normal_Radiation(self, n): float
+ self.Calculate_Angle_of_Incidence(self): float
+ self.Calculate_Extraterrestrial_Horizontal_Radiation(self, n, t_c, G): float
+ self.Calculate_Beam_Ratio(self): float
+ self.Calculate_Clearness_Index(self, G): float
+ self.Calculate_Diffuse_Beam_Radiation(self, G): float
+ self.Calculate_Anistropy_Index(self): float
+ self.Calculate_Horizon_Factor(self, G): float
+ self.Calculate_Incident_Radiation(self, G): float
+ self.Calculate_Power_Output(self): float
```

```
ACSE9.StorageModel.StorageArray
+ self.eff_round: float
+ self.v_nominal: float
+ self.count: int
+ self.dt: float
+ self.q_nominal: float
+ self.SoC: float
+ self.SoC_min: float
+ self.i_charge: float
+ self.i_discharge: float
+ self.Power_Output: float

+ self.Caculate_Charge_Discharge(self, P_solar, P_wind, P_load)
```

Figure 2.2:
Class Diagrams of Power Output Model

Solar Model

The ACSE9 tool solar PV power output model is considerably more involved than the wind model. The model takes just one input (other than the PV module physical specifications) - a resource profile of irradiance (in kW.m^{-2}) at every time-step. The solar PV model requires several separate functions to calculate the power output at the current timestep. These functions can be broadly divided into four main processes: (1) calculation of solar orientation angles, (2) calculation of incident radiation, (3) calculation of cell temperature, and (4) calculation of the solar PV power output. Note, the Author undertook significant work to design a full solar power output model workflow. The theoretical basis and function definitions related to the model are included in Appendix 1.

ACSE9.SolarModel Algorithm (time-step)

```
Inputs  time_zone, latitude, longitude, slope,
        azimuth, dt, Module_Capacity,
        Module_Count, albedo, Derating_Factor,
        irradiance profile representative of the
        (latitude, longitude) location.
1:      Define array inside class:
        `ACSE9.SolarModel.SolarArray`
2:      data in
        irradiance at current time-step
3:      Calculate orientation angles[5]
4:      Calculate extra-terrestrial radiation[6]
5:      if (irradiance > extra-terrestrial radiation)
        & (hour angle outside of sunrise/sunset)
        then Author's sunset exception activated:
          set extra-terrestrial radiation arbitrarily
        lolarger than (negligible) irradiance.
6:      Calculate incident radiation[7]
6:      Calculate cell temperature[8]
7:      Calculate self.Power_Output[9]
8:      end
```

Storage Model

Compared with other power output models, 3.14 docs provides less information regarding the power output model for what it names the 'idealized battery'. The storage power output and State-of-Charge (SoC) model developed by the author for the ACSE9 tool uses the same attributes as defined in HOMER Pro's 'idealized battery' but employs an original charging decision algorithm.

ACSE9.StorageModel Algorithm (time-step)

```
Inputs  eff_round, v_nominal, count, dt,
        q_nominal, SoC_min, i_discharge,
        i_charge, initial SoC
1:      Define array inside class:
        `ACSE9.StorageModel.StorageArray`
2:      data in
        PV power generated during time-step
        wind power generated during time-step
        electric load to satisfy over time-step
3:      Calculate, delta = load - generation
4:      if delta > 0 then discharge:
5:        Account for efficiency losses
6:        Calculate required current, I
7:        Calculate current through each branch, i*
*         (Assuming parallel configuration)
8:        Calculate charge consumption, Q
9:        Calculate change in SoC, SoC_delta
10:       if self.SoC + SoC_delta < self.SoC_min
          then not enough charge:
11:         self.Power_Output = 0.0
12:       else then discharge:
13:         if np.abs(i) > self.i_discharge
            then discharge current too large:
14:           self.Power_Output = 0.0
15:         else then discharge:
16:           self.Power_Output = I*self_v_nom
17:           self.SoC += SoC_delta
18:     if delta < 0 then charge:
19:       repeat lines 5-9
20:       if self.SoC == 100
          then cannot accept any more charge:
21:         self.Power_Output = 0.0
22:         skip to line 35
23:       if self.SoC + SoC_delta > 100
          then charge must be curtailed:
25:         calculate charge left to full charge,
            Q = 100 – self.SoC
26:         calculate current required to meet Q,
            I = Q / self.dt
27:         calculate current through each branch, i
28:         if np.abs(i) > self.i_charge
            then charge current too large, curtail:
29:           calculate maximum charge current,
              I = self.i_charge * count
29:           repeat lines 26, 16
30:           self.SoC+= Q/(q_nom*count)
31:         if np.abs(i) <= self.i_charge
            then charge curtailed amount
32:           repeat lines 16, 17
33:       if self.SoC + SoC_delta == 100
          then charge array:
34          repeat lines 28-31
35      end
```

c. FINANCIAL MODEL

The financial model is a simple lifecycle discounted cash-flow model which considers capital, operations and maintenance and replacement costs, and the revenue from salvage at the end of the project lifetime. The model is discounted to include the effects of inflation and the nominal discount rate[10] (rate at which the project can borrow money).

(2) Equation A.1 (3) Equation A.2 (4) Equation A.3 (5) Equations A.4 – A.9
(6) Equations A.10-A.12 (7) Equation A.14 (8) Equation A.15 (9) Equation A.17 (10) An inflation rate of 2% and discount rate of 8% were used in the analyses of this project.

d. VERIFICATION AND VALIDATION ROUTINES

Verification

Verification of complex functions used in the ACSE9 model was achieved by comparison with existing literature and the inclusion of these comparisons into a continuous integration infrastructure.

Comparison inputs and results were found for the calculations of solar declination (Duffie and Beckman, 1991), solar time (National Oceanic and Atmospheric Administration, 2021), solar hour angle (Honsberg and Bowden, 2021), angle of incidence, extra-terrestrial normal radiation and extra-terrestrial horizontal radiation (Duffie and Beckman, 1991). These comparison cases were included as parameters to test cases written using the `pytest` (krekel and dev-team, 2021) testing framework, which were in turn included in a GitHub Actions workflow to provide a level of automated testing to the continuous integration framework. Unit tests were also included for the verification of `ACSE9.FinancialModel`. The test cases for financial model verification were taken from 3.14 docs.

Validation

The advantage of the ACSE9 tool is that it provides a platform upon which the functionality of HOMER Pro-based models can be improved. Thus, it was essential to validate all models included in the baseline ACSE9 tool against results generated from HOMER Pro.

The power output models introduced in Sections 2.a and 2.b were validated against HOMER Pro by constraining the search-space to a single known-feasible HPP configuration for a given load profile. Then, by running the power output models at three distinct geographical locations. The known-feasible HPP configuration was 100-Cells, 25-WTGs, and 25-Modules, with a total plant capacity of 200kW. The locations of each validation case were chosen because a variety of solar dynamics would be experienced by the solar model. This variety was achieved by selecting locations corresponding to the Northern Nemisphere (NH) (Brighton, United Kingdom), Equator (Eq) (Lake Turkana, Kenya), and Southern Hemisphere (Melbourne, Australia). Table 2.1 details the results of the power model validation, Figure 2.3 illustrates the wind, solar and storage power output model results, Figure 2.4 illustrates the delta errors in each set of validation results (HOMER – ACSE9).

Table 2.1: Power Model Validation Results

Test Case	Minimum Power Output / kW (HOMER/ ACSE9)	Maximum Power Output / kW (HOMER/ ACSE9)	Average Power Output / kW (HOMER/ ACSE9)	Average Error
NH Wind	0.00/ 0.00	75.0/ 75.0	31.2/ 31.2	0.031 %
Eq Wind	0.00/ 0.00	75.0/ 75.0	33.3/ 33.3	0.043 %
SH Wind	0.00/ 0.00	75.0/ 75.0	21.0/ 21.0	0.003 %
NH Solar	0.00/ 0.00	25.4/ 22.3	3.11/ 2.72	12.6 %
Eq Solar	0.00/ 0.00	24.4/ 24.0	4.98/ 4.98	0.161 %
SH Solar	0.00/ 0.00	24.6/ 24.5	3.82/ 3.53	7.46 %
NH Storage	-5.33/ -5.01	1.60/ 1.52	-0.00253/ -0.00237	6.35 %
Eq Storage	-71.3/ -70.7	23.9/ 22.7	-0.103/ -0.0950	7.82 %
SH Storage	-4.87/ -4.62	2.20/ 2.09	-0.00472/ -0.00436	7.44 %

Notable, are the relatively large validation errors for NH and SH solar, and all of the storage test cases. The solar model error is highly related to the 'sunset exemption' implemented by the Author (see Appendix 1). If the sunset exemption is edited such that the extra-terrestrial horizontal radiation is arbitrarily set to 3 times the irradiance resource data, then the NH solar and SH solar validation errors drop to 2.85% and 1.59% respectively. However, this comes at a cost of the peak delta error (Figure 2.4), which rises from around 3 MW (NH) and 1.5 MW (SH) to -15 MW (NH) and -12 MW (SH). With regards to the storage model validation errors, as noted in Section 2.b, `ACSE9.StorageModel` is an entirely original model, with an original charge/discharge algorithm developed in lieu of adequate documentation provided by HOMER. Unfortunately, the theoretical basis of the remaining validation errors across these solar and storage validation cases could not be identified during this project and is a topic for future work.

GitHub repository: https://github.com/acse-2020/acse2020-acse9-finalreport-acse-tc20.git

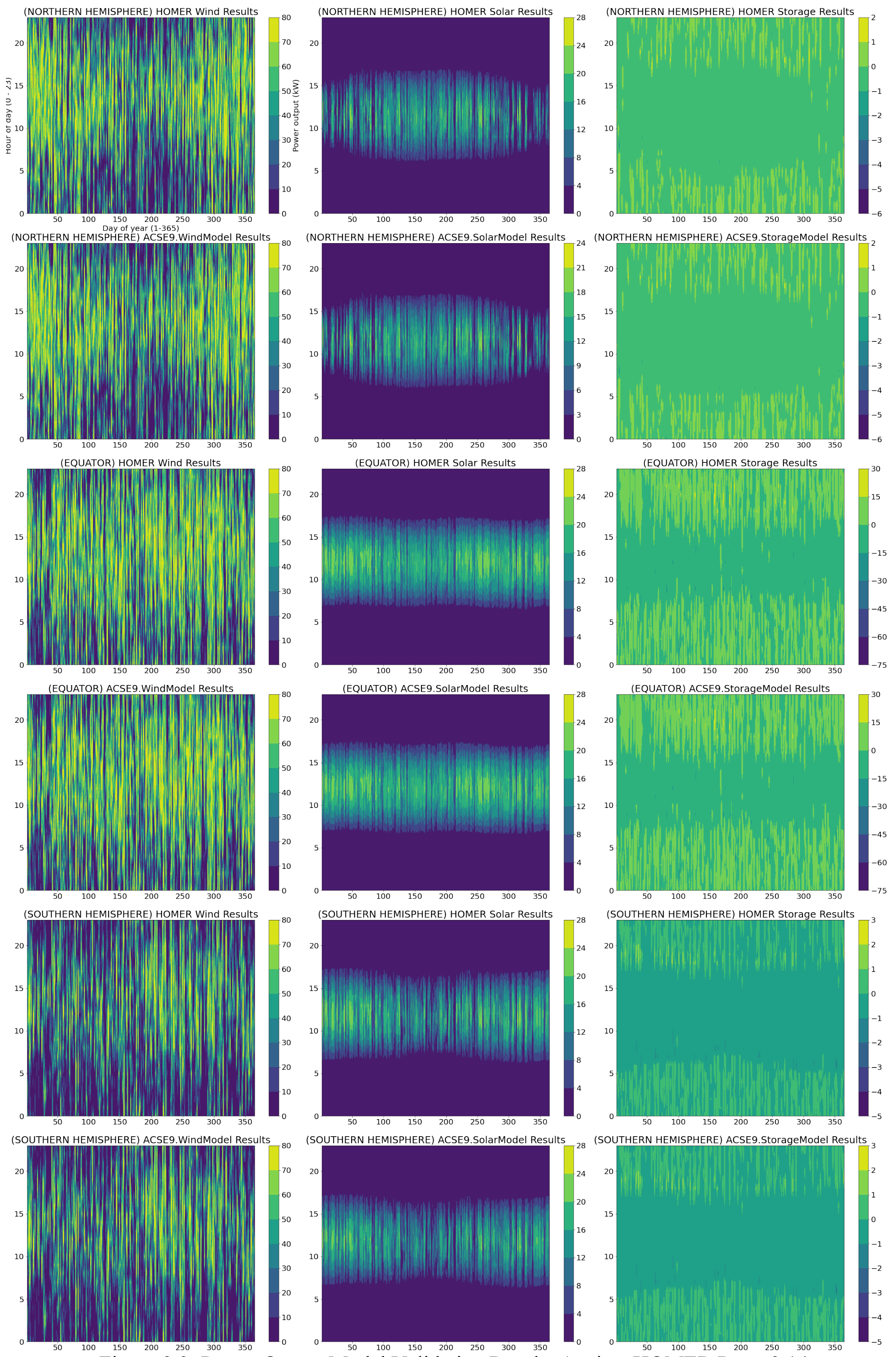


Figure 2.3: Power Output Model Validation Results Against HOMER Pro v.3.14

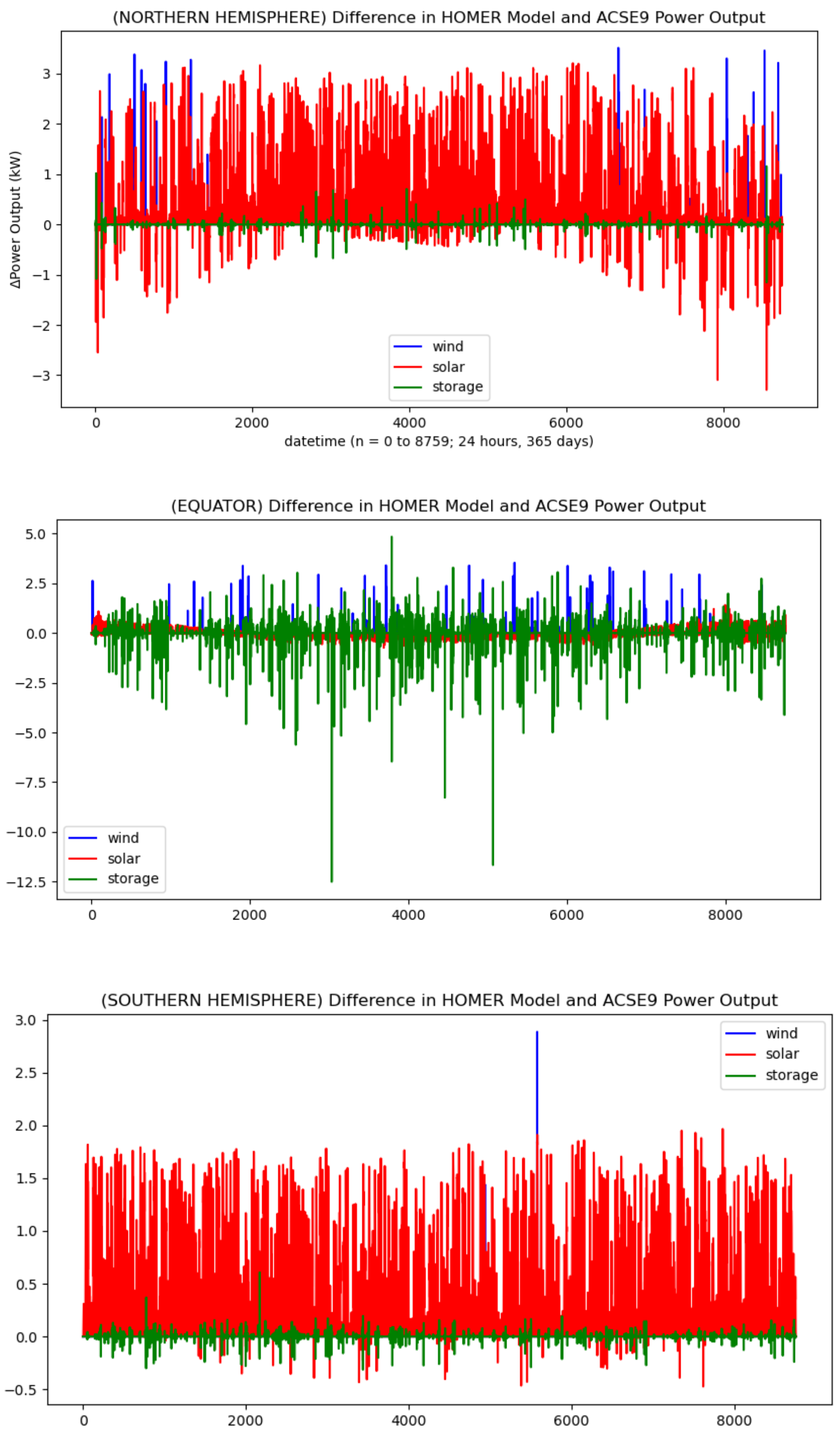


Figure 2.4: Delta errors in each set of power output validation results, over all time-steps

e. ALTERNATIVE OPTIMISATION METHODS

Thus far, this report has discussed an architecture which relies on exhaustive search. The purpose of the ACSE9 tool is to provide a platform for an advanced optimisation technique which enables more accurate, but more computationally expensive, power output models. One state-of-the-art method for optimising HPPs was considered in the development of ACSE9 - the evolutionary algorithm, differential evolution. Anoune et al (2018) provided a literature review of research which has used this technique for the purpose of HPP sizing. Particularly relevant is Koutroulis et al (2006) who used a genetic algorithm to optimise based on plant cost with a load fulfilment feasibility requirement.

The models developed in the ACSE9 tool were designed to integrate well with other pre-packaged optimisation methods, a functionality not offered by HOMER Pro. Notably, the ACSE9 tool optimisation process can be performed by SciPy's (Virtanen, 2020) differential evolution and brute global optimisation algorithms – referred to as the SciPy optimisers hereafter. The differential evolution function provided by SciPy is based on the global optimisation technique developed by Storn and Price (1997).

The exhaustive search algorithm templated in main.py and described in Section 2.a is a discrete search in which HPP configurations are designed based on integer counts of components. There exist integer-variable global optimisation algorithms under the larger umbrella of Mixed Integer Optimisations (see Nemhauser and Wolsey, 1999), however it was not possible to integrate these within the ACSE9 tool due to the constraints of this report. The SciPy optimisers are continuous (i.e., floating point-based) and thus take significantly more time to run than the discrete exhaustive search performed in main.py and by HOMER Pro. Therefore, comparison between the continuous SciPy optimisers and exhaustive search must be performed by utilising a continuous exhaustive search algorithm (`scipy.optimize.brute`).

3. RESULTS AND DISCUSSION

This section discusses results from the simulation of several HPP scenarios using the ACSE9 tool. The simulations were run on 1.7 GHz 6-Core Intel Core i5 and 8 GB RAM.

Using the equator test case as an input, both HOMER Pro and the ACSE9 tool discrete exhaustive search methods were run with a search space from 0 to 100 of each component (WTGs, solar PV modules, li-ion cells). Initially, this search was run at a resolution of 1 component (i.e. 0, 1, 2, 3, …, 98, 99, 100). HOMER Pro completed in 2 hours 32 minutes, the ACSE9 tool ran for around 26 hours, but failed due to memory shortage on the machine. To rectify this issue, the Author considered a search resolution of 1 infeasible for running multiple tests, and the resolution was increased to 10 components (i.e. 0, 10, 20, 30, …, 80, 90, 100). HOMER Pro finished its exhaustive search at this new resolution in 11 seconds, finding an optimal HPP configuration of 10-WTGs, 30-modules and 100-cells at an NPC of $404,499. The ACSE9 tool finished with a runtime of 2 minutes 48 seconds, finding an optimal solution of 10-WTGs, 30-modules and 90-cells at an NPC of $395,757. The author contends that the difference in optimal solution may be accounted for in the 7.82% validation error for the storage model presented in Table 2.1. If the storage model error did account for the difference in optimal HPP configuration, then HOMER Pro would find an optimal configuration of 10-WTGs, 30-modules and 98-cells, with the additional 8 cells accounting for the circa 8% error. Running HOMER Pro on this restricted search space yielded this feasible solution with an NPC of $402,762. HOMER Pro utilises parallelisation in order to speed up runtime. Making the optimistic assumption that parallelisation would reduce the runtime proportionally to the number of increased processors, it is reasonable to assume that the runtime of the ACSE9 tool could be reduced to 28 seconds (2 minutes 48 seconds divided by 6 CPUs), thus bringing the runtime of the ACSE9 tool in line with HOMER pro, which runs in 11 seconds.

Again, using the equator test case inputs, the SciPy differential evolution optimisation function was run using the ACSE9 tool. A maximum capacity shortage fraction of 1% was used as a constraint for feasibility, the NPC was used as the objective, and a tolerance of $1000 was used as a convergence stopping point. The differential evolution function converged in 1 hour 40 minutes, finding an optimal solution of 4-WTGs, 39-modules and 100-cells with an

NPC of $295,944. Accounting for the circa 8% storage model error and running over a rejected workspace, HOMER Pro confirmed the validity of this result by finding a feasible solution of 4-WTGs, 39-modules and 112-cells with an NPC of $304,729. The longer runtime of the differential evolution function is less significant when considering the continuous nature of the optimisation.

The ACSE9 tool exhaustive search is a discrete method which was run with a resolution of 10 components. The differential evolution method is a continuous method which was run over the full 0 to 100 component search space. As noted, the ACSE9 tool discrete exhaustive method search with a resolution of 1 component was computationally prohibitive. Thus, it is reasonable to assume that a discrete differential evolution would traverse a full 0 to 100 component search space with a resolution 1 component faster than discrete exhaustive search. Because a continuous exhaustive search cannot be run due to memory limits and runtime, it is difficult to draw a comparison between HOMER Pro discrete exhaustive search and the ACSE9 tool differential evolution runtime. Making the generous assumption that without memory constraints, the ACSE9 tool exhaustive search would be able to traverse a 0 to 100 component search space with a resolution of 1 component in 26 hours (the time of failure for discrete exhaustive search), then the differential evolution method can be assumed to be at least 16 times faster than exhaustive search. Since HOMER Pro was around 15 times faster than ACSE9 tool (11 seconds versus 2 minutes 28 seconds seconds) discrete exhaustive search, the differential evolution method in conjunction with the ACSE9 tool is assumed to be almost equal in runtime to HOMER Pro. With parallelisation, the ACSE9 tool using differential evolution would likely be significantly faster than HOMER Pro discrete exhaustive search.

To summarise, continuous exhaustive search (or discrete with a resolution of 1 component) should, by definition, be able to find a global minimum solution. But, this was not demonstrated as the technique was computationally prohibitive. Instead, both the ACSE9 tool and HOMER Pro discrete exhaustive search optimisations were run with a resolution of 10 components. The results of each of these methods were shown to be comparable after adjusting for validation error in the storage model. Next, SciPy’s differential evolution optimiser was used. This optimiser is a continuous method and found a better (25% cheaper NPC) optimal solution than discrete exhaustive search. If differential evolution were made discrete, and the operations of the ACSE9 tool were parallelised, then the ACSE9 tool and differential evolution combination would be better than the industry standard with regards to both results and runtime.

4. SUMMARY

This report introduces the ACSE9 tool, a power output and financial modelling platform which can be used to optimise the component sizing of a HPP. The base models and exhaustive search optimisation workflow are based on the industry standard software for HPP sizing, HOMER Pro (v3.14).

The ACSE9 tool is validated against HOMER Pro across a range of geographical locations, such that it is tested robustly against several solar dynamics in the northern hemisphere, equator and southern hemisphere. To optimise a HPP plant configuration, the ACSE9 tool power output models are fed representative wind speed, irradiance and load profiles, and then run for different WTG, solar PV module, and Li-ion cell component counts within a user-defined search space. Every possible combination of each component search space was considered as a configuration, and the feasibility of each configuration was checked by comparing its CSF to a user-defined maximum. If a configuration is considered feasible, then its NPC is calculated by the ACSE9 tool financial model. The optimal HPP configuration has the lowest NPC.

When the equator validation case was run using discrete exhaustive search over a reduced search-space with 10^3 possible configurations, the ACSE9 tool finds an optimal combination of 10-WTGs, 30-modules and 90-cells and runs in 2 minutes 48 seconds. When run on the same set up, HOMER Pro finds an optimal combination of 10-WTGs, 30-modules and 100-cells and runs in 11 seconds. The discrepancy between the two results is explained by the validation error presented in Table 2.1. If the equator case validation error is accounted for in the ACSE9 tool optimal combination, it successfully matches the HOMER Pro result. The discrepancy in runtime is likely explained by parallelisation and other proprietary timing optimisations implemented by HOMER Pro.

Beyond exhaustive search, this report sought to explore whether alternative global optimisation techniques, such as those presented by Anoune et al (2018), can be integrated in the open-source ACSE9 tool platform, and whether these alternative techniques can provide improvements in runtime and/or optimal solution. The power output, feasibility, and financial model workflow of the ACSE9 tool was integrated with the differential evolution optimiser provided in the optimize module of the SciPy package. Differential evolution on the same equator test case yielded a 25% cheaper NPC optimal solution of 4-WTGs, 39-modules and 100-cells and ram in 1 hour 40 minutes. Again, the discrepancy between these solutions and the HOMER Pro results is explained by the validation error.

Further to the additional functionality it provides, the ACSE9 tool provides several other benefits when compared to HOMER Pro. As an open-source, non-UI-based software, the ACSE9 tool allows for efficient and exploratory scripting. Scripting is a useful functionality for conducting research over a variety of search-spaces, geographical locations, and component data. Moreover, scripting combined with open access to source code allows for user investigation of any desired sensitivity analysis. The ACSE9 tool also includes pre-made data import and plotting functions which restrict input datatypes to avoid user-error; an original storage charge-discharge algorithm which was developed in lieu of adequate HOMER Pro documentation; and documented explanations of the theoretical background and assumptions made in the development of the software (Appendix 1).

Future Work

The architecture of the ACSE9 tool provides a useful platform for the development of a better-than industry standard software. The first step to improve the tool would be to reduce all validation errors presented in Table 2.1 to near-zero, or, as noted below, to improve and/or replace the HOMER Pro models with higher-accuracy alternatives. When the results are considered exactly equal to HOMER Pro, the run time of the ACSE9 tool could be improved with parallelisation. Since the ACSE9 tool in conjunction with SciPy's differential evolution optimisation is already comparable in runtime to HOMER Pro discrete exhaustive search, parallelisation would give a significant improvement on HPP optimisation runtime compared with the industry standard.

Using the opportunity presented by the decreased runtime (i.e., reduced computational load), the ACSE9 tool should seek to increase the complexity of the power output models used in its simulation. Opportunities from power output model improvement include, but are not limited to, a dynamic consideration of WTG wake effects, a dynamic model for PV module and li-ion cell aging, and the development of increased complexity storage models which consider, for example, SoC versus nominal voltage effects. Development could also continue by comparing to industry standard; developing higher resolution power output models which seek to match the performance of industry standard energy yield analysis software such as PVSyst and WaSP.

Future work should also include the inclusion and analysis of additional global optimisation methods, such as those presented by Anoune et al (2018). Notably, well-designed multi-objective optimisation methods would allow for optimisation not only based on NPC, but also added constraints and/or objectives such as lifecycle emissions and grid stability.

Finally, regarding the grid placement of the optimised HPP, it would be useful to develop a power trading revenue model which would reflect a grid-connected plant. HOMER Energy provides the industry standard for grid connected HPP sizing optimisation in its HOMER Grid software.

REFERENCES

Anoune, K., Bouya, M., Astito, A. & Abdellah, A.B. 2018. Sizing methods and optimization techniques for PV-wind based hybrid renewable energy system: A review. Renewable and Sustainable Energy Reviews. [Online] 93, 652–673. Available from: doi:10.1016/j.rser.2018.05.032.

Climate Change Committee. 2020. The Sixth Carbon Budget Electricity generation. [online] Available at: <https://www.theccc.org.uk/wp-content/uploads/2020/12/Sector-summary-Electricity-generation.pdf> [Accessed 8 August 2021].

Duffie, J., Beckman, W. 1991. Solar engineering of thermal processes. Hoboken, N.J.: John Wiley & Sons.

Ekren O, Ekren BY. Size optimization of a PV/wind hybrid energy conversion system with battery storage using simulated annealing. Appl Energy 2010;87(2):592–8.

HOMER Energy. 2020. Net Present Cost. [online] www.homerenergy.com. Available at: <https://www.homerenergy.com/products/pro/docs/latest/net_present_cost.html> [Accessed 8 Aug. 2021].

Honsberg, C. and Bowden, S., 2021. Solar Time | PVEducation. [online] Pveducation.org. Available at: <https://www.pveducation.org/pvcdrom/properties-of-sunlight/solar-time> [Accessed 16 August 2021].

Koutroulis E, Kolokotsa D, Potirakis A, Kalaitzakis K. Methodology for optimal sizing of stand-alone photovoltaic/wind-generator systems using genetic algorithms. Sol Energy 2006;80(9):1072–88.

krekel, h. and dev-team, p., 2021. pytest: helps you write better programs — pytest documentation. [online] Docs.pytest.org. Available at: <https://docs.pytest.org/en/6.2.x/> [Accessed 16 August 2021].

Monforti F, Huld T, Bodis K, Vitali L, D'Isodoro M, Lacal-Arantegui R. Assessing complementarity of wind and solar resources for energy production in Italy. A Monte Carlo Approach. Renewable Energy 2014;63: 576-586.

National Oceanic and Atmospheric Administration, 2021. Sun Calculator. [online] Available at: <http://geoastro.de/astro/suncalc/index.htm> [Accessed 16 August 2021].

Nemhauser, G., Wolsey, L. 1988. Integer and combinatorial optimization. Wiley.

Paska, J., Biczel, P. and Kłos, M. 2009. Hybrid power systems – An effective way of utilising primary energy sources. Renewable Energy, 34(11), pp.2414–2421.

Schlömer S., T. Bruckner, L. Fulton, E. Hertwich, A. McKinnon, D. Perczyk, J. Roy, R. Schaeffer, R. Sims, P. Smith, and R. Wiser. 2014. Annex III: Technology-specific cost and performance parameters. In: Climate Change 2014: Mitigation of Climate Change. Contribution of Working Group III to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change. Cambridge University Press, Cambridge, United Kingdom and New York, NY, USA.

Storn, R., Price, K. 1997. Differential Evolution - a Simple and Efficient Heuristic for Global Optimization over Continuous Spaces, Journal of Global Optimization. 11, 341 - 359.

Tsallis C, Stariolo DA. 1996. Generalized Simulated Annealing. Physica A, 233, 395-406.

Virtanen, P, Gommers, R., Oliphant, T., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S., Brett, M., Wilson, J., Millman, J., Mayorov, N., Nelson, A., Jones, E., Kern, R., Larson, E., Carey, C., Polat, I., Feng, Y., Moore, E., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E., Harris, C., Archibald, A., Ribeiro, A., Pedregosa, F., van Mulbregt, P., and SciPy 1.0 Contributors. (2020) SciPy 1.0: Fundamental Algorithms for

Scientific Computing in Python. Nature Methods, 17(3), 261-272.

World Bank. 2021. Renewable electricity output (% of total electricity output) | Data. [online] Available at: <https://data.worldbank.org/indicator/EG.ELC.RNEW.ZS> [Accessed 8 August 2021].

APPENDIX 1

Theoretical documentation of power output models

A. POWER OUTPUT MODELS

The ACSE9 tool contains three power output models which simulate the power output at every timestep of an array of generation or storage components. ACSE9.WindModel models an array of Wind Turbine Generators (WTGs), ACSE9.SolarModel, models an array of solar photovoltaic (solar PV) modules, and ACSE9.StorageModel models a collection of Li-ion cells. All of the power output models presented in this report, and available in the ACSE9 tool (as of v.0.1), are based off of the calculations used by HOMER Pro v3.14 – the off-grid HPP sizing tool sold by HOMER Energy. These calculations can be found at <https://www.homerenergy.com/products/pro/docs/3.14/index.html>: Any reference made to the equations found within this documentation will be denoted as (3.14 docs) for the remainder of this paper.

The power output models are defined as python classes, namely `ACSE9.WindModel.WindArray`, `ACSE9.SolarModel.SolarArray`, and `ACSE9.StorageModel.StorageArray`. Defining the models as such allows for efficient scripting, where the class attributes and methods associated with each model can be varied for any desired sensitivity analyses.

ii. ACSE9.WindModel

The ACSE9 tool wind power output model is a relatively simple power-curve-based module. The model takes two inputs (other than the WTG physical specifications): A resource profile of wind-speed (in m.s^{-1}) at anemometer height (in m), at every time-step for which the model is to be run; and a representative power curve for the WTG which is to be modelled. A power curve is a dataset which contains manufacturer data on how the power output (in kW) of a single WTG varies with the wind speed at hub-height (in m), at a Standard Air Density (SD), ρ_0, of 1.225 kg.m^{-3}.

The calculation of the power output of the WTG array at each time step has three stages. First, the wind speed at WTG hub-height is calculated by adjusting the wind speed at anemometer height, taken from the imported resource profile. The adjustment is made by applying the logarithmic law given in Equation A.1 (3.14 docs).

$$U_{hub} = U_{anem}\left(\frac{\ln\frac{z_{hub}}{z_0}}{\ln\frac{z_{anem}}{z_0}}\right) \qquad \text{(A.1)}$$

Where:
U_{hub} is the wind speed at WTG hub-height, in m.s^{-1}; U_{anem} is the wind speed at anemometer height, in m.s^{-1}; z_{hub} is the height of the WTG hub, in m; z_{anem} is the height of the anemometer, in m; and z_0 is the surface roughness in m.

After the wind speed at hub-height has been calculated, the expected SD power output of the WTG array at the current time-step is calculated performing linear interpolation the imported power curve. Equation A.2 (adapted by Author, from 3.14 docs) describes this process.

$$P_{WA,SD} = n_{WTG}\left(P_{WTG,SD,1} + \left(\frac{P_{WTG,SD,2}-P_{WTG,SD,1}}{U_{hub,2}-U_{hub,1}}\right)\left(U_{hub} - U_{hub,1}\right)\right)$$

(A.2)

Where:
$P_{WA,SD}$ is the power of the WTG array at SD, in kW; n_{WTG} is the number of WTGs in the array. Given U_{hub} from Equation 2.1, upper and lower interpolation datapoints are read from the imported WTG power curve. $U_{hub,2}$ and $U_{hub,1}$ are the upper and lower datapoints, respectively, for hub-height wind speed, in m.s^{-1}; $P_{WTG,SD,2}$ and $P_{WTG,SD,1}$ are the upper and lower datapoints, respectively, for WTG power at SD, in kW.

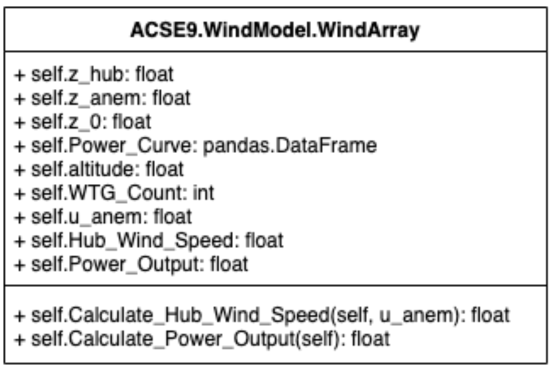
Finally, the predicted WTG power output is calculated by adjusting the result from Equation A.2 for the air density at the WTG array location. Air density is assumed to be a function of altitude only. Since SD is defined at an altitude of zero meters above sea level; if the altitude of the array is defined as zero, the air density adjustment applied below, in Equation A.3 (3.14 docs), will have no effect.

$$\begin{aligned} P_{AW} &= P_{AW,SD}\left(\frac{\rho}{\rho_0}\right) \\ &= P_{AW,SD}\left(1 - \frac{Bz}{T_0}\right)^{\frac{g}{RB}}\left(\frac{T_0}{T_0 - Bz}\right) \qquad \text{(A.3)} \end{aligned}$$

Where:

P_{AW} is the Power of the WTG array, in kW, adjusted for air density; ρ is the air density, in kg.m^{-3}, at the array altitude; B is the lapse rate, equal to 0.0065 K.m^{-1}; z is the altitude of the WTG array, in m; T_0 is the temperature at Standard Temperature Pressure (STP), equal to 288.16 K; g, is acceleration due to gravity, equal to 9.81 m.s^{-2}; R is the gas constant, equal to 287 J.kg^{-}1K^{-1}.

The wind array class defined in `ACSE9.WindModel.WindArray` has two class methods, which are run by calling `ACSE9.RunWindModel` to find the power output of the WTG array at the every time-step over the length of the imported resource profile. The class diagram for the ACSE9 tool wind power output model is illustrated in Figure A.1.



ACSE9.WindModel.WindArray
+ self.z_hub: float + self.z_anem: float + self.z_0: float + self.Power_Curve: pandas.DataFrame + self.altitude: float + self.WTG_Count: int + self.u_anem: float + self.Hub_Wind_Speed: float + self.Power_Output: float
+ self.Calculate_Hub_Wind_Speed(self, u_anem): float + self.Calculate_Power_Output(self): float

Figure A.1: Class Diagram for `ACSE9.WindModel.WindArray`

ii. ACSE9.SolarModel

The ACSE9 tool solar PV power output model is considerably more involved than the wind model. The model takes one input (other than the PV module physical specifications): A resource profile of irradiance (in kW.m^{-2}) at every time-step for which the model is to be run. The solar PV model requires 17 separate functions to calculate the power output at the current timestep. These functions can be broadly divided into four main processes - the calculation of solar orientation angles, the calculation of incident radiation, the calculation of cell temperature, and the calculation of the solar PV power output.

Solar Orientation Angles

Before giving a detailed description of the modelling of solar PV power output, one must understand solar orientation angles.

Figure A.2 gives an ariel-view and side-view, of a photovoltaic panel's orientation and the associated solar angles.

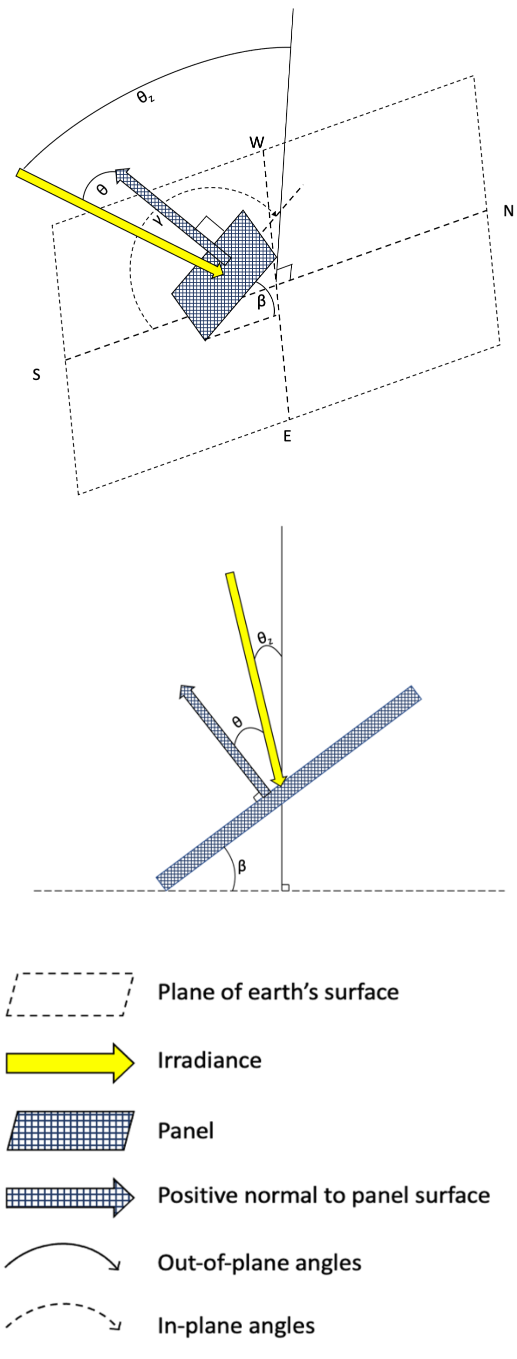


Figure A.2: Solar orientation angles

Where: θ is the angle of incidence - the angle between the direction of irradiance and the axis normal to the surface of the panel, in degrees; θ_z is the zenith angle - the angle between the axis normal to the earth and the irradiance, in degrees; β is the slope of the panel - the angle between the ground and the surface of the panel, in degrees; γ is the azimuth of the surface - the direction in which the panel faces (HOMER Pro convention is that due south is zero azimuth, and the azimuth is positive in the clockwise direction), in degrees. Other solar orientation angles, which are dependent on the latitude (ϕ) and longitude (λ) of the panel (but independent of the panel orientation) are discussed below.

The solar declination, δ, is the latitude, in degrees, at which the irradiance is normal to the earth's surface at solar noon. HOMER Pro uses Equation A.4 (3.14 docs) to calculate the solar declination, given the day of the year, n, from 1-365.

$$\delta = 23.45° \sin\left(360° \frac{284+n}{365}\right) \qquad \text{(A.4)}$$

The hour angle, ω, is an angle which describes the suns position in the sky with respect to time. The hour angle is equal to 0 ° at solar noon, and is based upon the solar time, t_s in hours. HOMER Pro uses Equation A.5 to calculate the solar time, and Equation A.6 to calculate the hour angle (3.14 docs).

$$t_s = t_c + \frac{\lambda}{15° hr^{-1}} - Z_c + E \qquad \text{(A.5)}$$
$$\omega = (t_c - 12hr) * 15° hr^{-1} \qquad \text{(A.6)}$$

Where:
t_c is the civil time, in hours; Z_c is the time zone in hours east of GMT; and E is the equation of time, in hours.

The equation of time is used to account for the effects of obliquity and eccentricity associated with the elliptic shape of the earth's orbit and is calculated as shown in Equation A.7 (3.14 docs).

$$\begin{aligned} E = 3.82(&0.000075 \\ &+ 0.001868\cos(B) \\ &- 0.032077\sin(B) \\ &-0.014615\cos(2B) \\ &-0.04089\sin(2B)) \end{aligned} \qquad \text{(A.7)}$$

Where B is shorthand for the term described in Equation A.8.

$$B = 360° \frac{n-1}{365} \qquad \text{(A.8)}$$

Using the solar orientation angles described above, the angle of incidence is calculated using Equation A.9 (3.14 docs).

$$\begin{aligned} \cos\theta = &\sin\delta \sin\phi \cos\beta \\ &- \sin\delta \cos\phi \sin\beta \cos\gamma \\ &+ \cos\delta \cos\phi \cos\beta \cos\omega \\ &+ \cos\delta \sin\phi \sin\beta \cos\gamma \cos\omega \\ &+ \cos\delta \sin\beta \sin\gamma \sin\omega \end{aligned} \qquad \text{(A.9)}$$

Note that since it is the angle between the axis normal to the earth and the irradiance, the zenith angle can be found from Equation A.9 by setting the slope to zero.

Incident Radiation

HOMER Pro accepts either monthly averages or time-series data for the global horizontal irradiance, $\bar{G}$ – which is the irradiance present post-atmosphere. This data can be supplied by the user or taken from NREL databases by specifying the location of the plant. While the $\bar{G}$ data are uploaded directly into HOMER Pro, the theoretical relationship between $\bar{G}$ and the extra-terrestrial (pre-atmosphere) horizontal radiation still must be understood for the purpose of resolving $\bar{G}$ into its beam and diffuse components.

The irradiance produced by the sun is considered constant; the solar constant, $\bar{G}_{sc}$, is equal to 1.367 kW.m^{-2}. The strength of this radiation reaching the top of earth's atmosphere depends on the position of the earth's elliptic orbit around the sun. Considering this orbit, HOMER Pro calculates the irradiance which strikes the top of the earth's atmosphere, termed the extra-terrestrial normal radiation, $\bar{G}_{on}$, as detailed in Equation A.10 (3.14 docs).

$$\bar{G}_{on} = \bar{G}_{sc}\left(1 + 0.033\cos\left(\frac{360n}{365}\right)\right) \qquad \text{(A.10)}$$

The component of $\bar{G}_{on}$ which passes through the atmosphere and is not reflected is termed the extra-terrestrial horizontal radiation, $\bar{G}_o$, and is calculate as detailed in Equation A.11 (3.14 docs), which can be expressed as equation A.12 (adapted by Author from 3.14 docs) by substituting an expression for the zenith angle taken from setting the slope to zero in Equation A.9 and integrating over the time-step.

$$\bar{G}_o = \bar{G}_{on}\cos(\theta_z) \qquad \text{(A.11)}$$

$$\bar{G}_o = \frac{12}{\pi}\bar{G}_{on}\left(\cos\phi\cos\delta\,(\sin\omega_2 - \sin\omega_1) + \frac{\pi(\omega_2-\omega_1)}{180°}\sin\phi\sin\delta\right)$$
(A.12)

It is worth noting here, HOMER Pro base their methods for calculating solar PV power output on those presented by Duffie and Beckman (1991). Duffie and Beckman do not present the method given in Equation A.12 to calculate $\bar{G}_o$. Instead, they calculate the average extra-terrestrial horizontal radiation over an entire day by integrating between the sunrise and sunset hour angles (ω_{rise} and ω_{set}, respectively). This is of course not appropriate for the HOMER Pro nor ACSE9 tool solar PV model, as we are concerned with time-step lengths of higher resolution than 24 hours. The use of Equation A.12 allows us to integrate at higher resolutions, but causes an issue with regards to sunrise and sunset:

When Equation A.12 is calculated at or near sunset, $\bar{G}_o$may reach a value of zero faster than the imported irradiance resource data. When $\bar{G}_o$ is zero, the global horizontal irradiance ($\bar{G}$) (which is incident below the earth's atmosphere, i.e., is not extra-terrestrial) should also equal zero. The discrepancy between $\bar{G}_o$ and $\bar{G}$ when modelling is likely to occur as $\bar{G}$ is usually data taken from solar radiation sensors, which are sensitive enough to continue measuring irradiance within many decimal places from zero. For example, a negligible value of $\bar{G}$ such as 0.001 kW/m^2 will still be read as non-zero. Since it is not physically possible for $\bar{G}$ to be larger than $\bar{G}_o$, the ACSE9 tool implements a 'sunset exception': If $\bar{G}$ > $\bar{G}_o$, and the hour angle is outside of daylight (ω_{rise}< ω < ω_{set}), then
$\bar{G}_o$ is artifically set higher than G.

Once $\bar{G}_o$has been calculated for the time-step, the beam and diffuse components of $\bar{G}$, $\bar{G}_b$ and $\bar{G}_d$ respectively, can be resolved. First, the clearness index, k_t, is calculated as the ratio of $\bar{G}$ to $\bar{G}_o$. Then, HOMER Pro uses the relationship given in Equation A.13 (3.14 docs) to calculate the ratio $\bar{G}_b$ to $\bar{G}$. $\bar{G}_d$ can then be found by simple subtraction.

$$\frac{\bar{G}_d}{\bar{G}} = \begin{cases} 1.0 - 0.09k_t & k_t \leq 0.22 \\ 0.9511 - 0.1604k_t + 4.388{k_t}^2 - 16.638{k_t}^3 + 12.336{k_t}^4 & 0.22 < k_t \leq 0.80 \\ 0.165 & k_t > 0.80 \end{cases}$$
(A.13)

Finally, the incident radiation on each PV module, $\bar{G}_T$, can be calculated using Equation A.14 (3.14 docs).

$$\bar{G}_T = (\bar{G}_b + \bar{G}_d A_i)R_b + \bar{G}_d(1 - A_i) * \left(\frac{1+\cos\beta}{2}\right) * \left(1 + f\sin^3\frac{\beta}{2}\right) + \bar{G}\rho_g\left(\frac{1-\cos\beta}{2}\right)$$
(A.14)

Where: A_i is the anisotropy index, which is the ratio of beam irradiance to extra-terrestrial horizontal radiation; R_b is the beam ratio, is the ratio of beam irradiance on the tilted surface of the panel to the beam irradiance on the horizontal surface of the ground (3.14 docs). R_b is calculated as the ratio of the cosine of the angle of incidence to the cosine of the zenith angle. By definition of the angle of incidence and the sunset hour angle, the cosine of the zenith angle is zero at sunset (i.e., a 90^o zenith angle). Thus, division by zero errors in the calculation of the beam ratio can occur at sunset if an exception is not made. The ACSE9 tool sets the beam ratio to 1 if the zenith angle is 90 degrees; and ρ_g is the albedo – a measure of the reflectance of the ground (%).

The function dependencies and full workflow implemented by the ACSE9 tool to calculate the irradiation incident at each time-step on the PV modules defined in `ACSE9.SolarModel.SolarArray` is illustrated in Figure A.3.

Cell Temperature

The temperature coefficient of power, α_p, is a measure of how much the power output of a PV module varies with its temperature. α_p is typically modelled as a linear factor and should be taken from manufacturers specifications. Some modelling suggestions for α_p are given by HOMER Pro, these are displayed in Table A.1 (3.14 docs) below:

Table A.1: Modelling Suggestions for Temperature Coefficient of Power

PV Module Type	α_p **(%.°C^{-1})**
Polycrystalline silicon	-0.48
Monocrystalline silicon	-0.46
Monocrystalline/amorphous silicon hybrid	-0.30
Thin film amorphous silicon	-0.20
Thin film CIS	-0.60

The efficiency of a PV array at its maximum power output, under Standard Test Conditions (STC) is denoted as $\eta_{mp,STC}$. STC are defined as, a cell temperature, $T_{c,STC}$, equal to 25 °C and an incident radiation, $\bar{G}_{T,STC}$, equal to 1 kW.m^{-2} (3.14 docs).

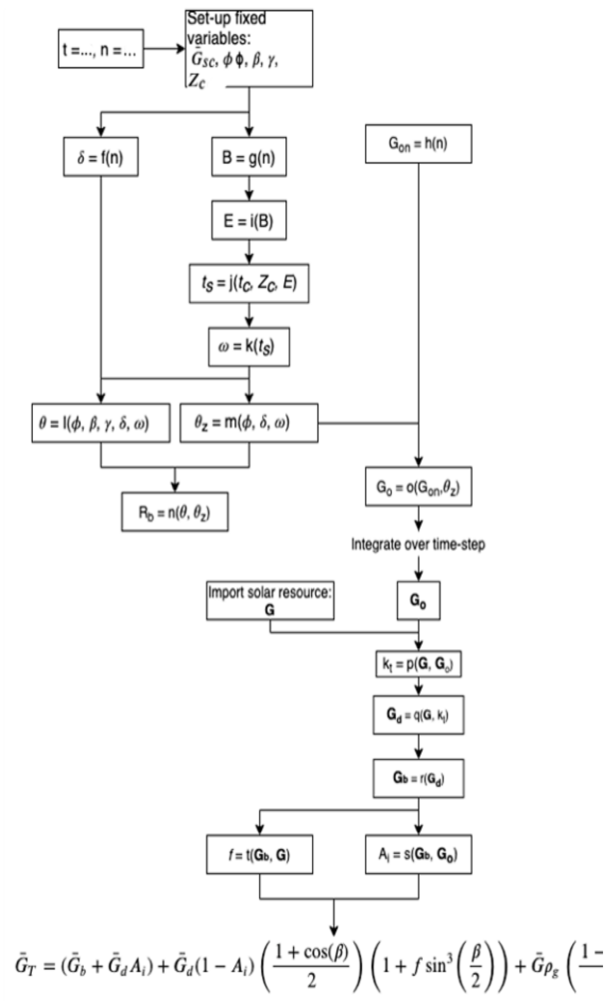

t =..., n =...
Set-up fixed variables: $\bar{G}_{SC}$, ϕ Φ, β, γ, Z_c
δ = f(n)
B = g(n)
G_{on} = h(n)
E = i(B)
t_S = j(t_C, Z_C, E)
ω = k(t_S)
θ = l(ϕ, β, γ, δ, ω)
θ_z = m(ϕ, δ, ω)
R_b = n(θ, θ_z)
G_o = o(G_{on}, θ_z)
Integrate over time-step
Import solar resource: G
G_o
k_t = p(G, G_o)
G_d = q(G, k_t)
G_b = r(G_d)
f = t(G_b, G)
A_i = s(G_b, G_o)
$\bar{G}_T = (\bar{G}_b + \bar{G}_d A_i) + \bar{G}_d(1 - A_i)\left(\frac{1+\cos(\beta)}{2}\right)\left(1 + f\sin^3\left(\frac{\beta}{2}\right)\right) + \bar{G}\rho_g\left(\frac{1-}{}\right.$

Figure A.3: Workflow and Dependencies for Incident Radiation Calculation

$\eta_{mp,STC}$ can be calculated using Equation 2.15 (3.14 docs), but HOMER Pro also provides some suggestions for modelling $\eta_{mp,STC}$, which are displayed in Table A.2 (3.14 docs).

$$\eta_{mp,STC} = \frac{Y_{PV}}{A_{PV}\bar{G}_{T,STC}} \qquad (2.15)$$

Where:
Y_{PV} is the rated capacity of the PV array, in kW; and A_{PV} is the surface area of the PV array.

Table A.2: Modelling Suggestions for Efficiency of a PV Array at Maximum Power Output, Under STC

PV Module Type	$\eta_{mp,STC}$ **(%)**
Polycrystalline silicon	13.0
Monocrystalline silicon	13.5
Monocrystalline/amorphous silicon hybrid	16.4
Thin film amorphous silicon	5.5
Thin film CIS	8.2

The Nominal Cell Operating Temperature (NOCT), $T_{c,NOCT}$, is the temperature at which the cell efficiency drops to zero, under test conditions of an ambient temperature, $T_{a,NOCT}$, equal to 20 °C, and incident radiation, $\bar{G}_{T,STC}$, equal to 0.8 kW.m^{-2} (3.14 docs). The NOCT should be taken from manufacturers specifications, but HOMER Pro suggests a range of 45-48 °C.

The cell temperature of the PV array, T_c , is calculated using Equation A.16 (3.14 docs).

$$T_c = \frac{T_a + (T_{c,NOCT} - T_{a,NOCT})\left(\frac{\bar{G}_T}{\bar{G}_{T,NOCT}}\right)\left(1 - \frac{\eta_{mp,STC}(\alpha_p T_{c,STC})}{0.9}\right)}{1 + (T_{c,NOCT} - T_{a,NOCT})\left(\frac{\bar{G}_T}{\bar{G}_{T,NOCT}}\right)\left(\frac{\alpha_p \eta_{mp,STC}}{0.9}\right)} \qquad (A.16)$$

Solar PV Power Output

The Solar PV power output at each timestep, P_{PV}, in kW, is calculated using Equation A.17 (3.14 docs).

$$P_{PV} = Y_{PV} f_{pv}\left(\frac{\bar{G}_T}{\bar{G}_{T,STC}}\right)\left(1 + \alpha_p\left(T_c - T_{c,STC}\right)\right) \qquad (A.17)$$

Where f_{pv} is termed the derating factor; a scaling factor used to account for reduced PV output in real-life conditions such as electrical losses, soiling and aging (3.14 docs).

The full workflow used by the ACSE9 tool for calculating the solar PV array power output at each timestep is illustrated in Figure A.4.

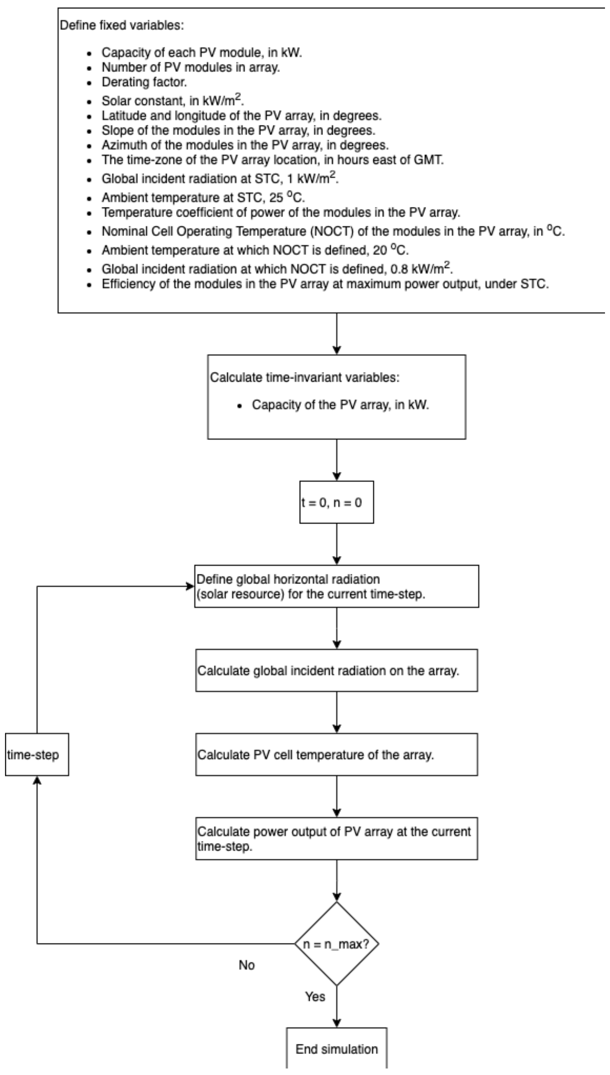
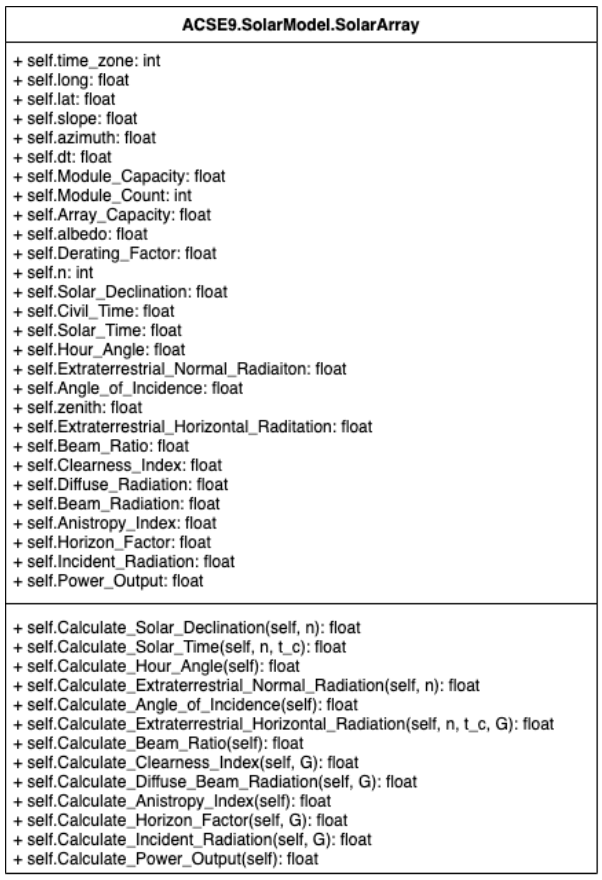

Define fixed variables:
Capacity of each PV module, in kW.
Number of PV modules in array.
Derating factor.
Solar constant, in kW/m².
Latitude and longitude of the PV array, in degrees.
Slope of the modules in the PV array, in degrees.
Azimuth of the modules in the PV array, in degrees.
The time-zone of the PV array location, in hours east of GMT.
Global incident radiation at STC, 1 kW/m².
Ambient temperature at STC, 25 °C.
Temperature coefficient of power of the modules in the PV array.
Nominal Cell Operating Temperature (NOCT) of the modules in the PV array, in °C.
Ambient temperature at which NOCT is defined, 20 °C.
Global incident radiation at which NOCT is defined, 0.8 kW/m².
Efficiency of the modules in the PV array at maximum power output, under STC.
Calculate time-invariant variables:
Capacity of the PV array, in kW.
t = 0, n = 0
Define global horizontal radiation (solar resource) for the current time-step.
Calculate global incident radiation on the array.
Calculate PV cell temperature of the array.
Calculate power output of PV array at the current time-step.
n = n_max?
No
Yes
time-step
End simulation

Figure A.4: ACSE9 Tool Full Workflow for Calculating the Solar PV Array Power Output at Each Time-Step

The solar array class defined in \`ACSE9.SolarModel.SolarArray\` has two class methods, which are run by calling \`ACSE9.RunSolarModel\` to find the power output of the solar PV array at the every time-step over the length of the imported resource profile. The class diagram for the ACSE9 tool solar PV power output model is illustrated in Figure A.5.



```
ACSE9.SolarModel.SolarArray

+ self.time_zone: int
+ self.long: float
+ self.lat: float
+ self.slope: float
+ self.azimuth: float
+ self.dt: float
+ self.Module_Capacity: float
+ self.Module_Count: int
+ self.Array_Capacity: float
+ self.albedo: float
+ self.Derating_Factor: float
+ self.n: int
+ self.Solar_Declination: float
+ self.Civil_Time: float
+ self.Solar_Time: float
+ self.Hour_Angle: float
+ self.Extraterrestrial_Normal_Radiaiton: float
+ self.Angle_of_Incidence: float
+ self.zenith: float
+ self.Extraterrestrial_Horizontal_Raditation: float
+ self.Beam_Ratio: float
+ self.Clearness_Index: float
+ self.Diffuse_Radiation: float
+ self.Beam_Radiation: float
+ self.Anistropy_Index: float
+ self.Horizon_Factor: float
+ self.Incident_Radiation: float
+ self.Power_Output: float

+ self.Calculate_Solar_Declination(self, n): float
+ self.Calculate_Solar_Time(self, n, t_c): float
+ self.Calculate_Hour_Angle(self): float
+ self.Calculate_Extraterrestrial_Normal_Radiation(self, n): float
+ self.Calculate_Angle_of_Incidence(self): float
+ self.Calculate_Extraterrestrial_Horizontal_Radiation(self, n, t_c, G): float
+ self.Calculate_Beam_Ratio(self): float
+ self.Calculate_Clearness_Index(self, G): float
+ self.Calculate_Diffuse_Beam_Radiation(self, G): float
+ self.Calculate_Anistropy_Index(self): float
+ self.Calculate_Horizon_Factor(self, G): float
+ self.Calculate_Incident_Radiation(self, G): float
+ self.Calculate_Power_Output(self): float
```

Figure A.1: Class Diagram for \`ACSE9.SolarModel.SolarArray\`

iii. ACSE9.StorageModel

Compared to the wind and solar PV power output models, HOMER Pro provides less information in 3.14 docs regarding the power output model for what it names the 'idealized battery'. The storage power output and State-of-Charge (SoC) model developed by the Author for the ACSE 9 tool uses the same attributes as defined in HOMER Pro's 'idealized battery', but uses an original charge/discharge decision algorithm.

The attributes defined in \`ACSE9.StorageModel.StorageArray\` are as follows: \`self.eff_round\` – the round trip efficiency of each storage cell (%); \`self.v_nom\` – the nominal voltage of each storage cell (V); \`self.count\` - the number of storage cells in the array; \`self.dt\` – the time-step size (hours); \`self.q_nom\` - the nominal capacity of each cell (Ah); \`self.SoC_min\` - the minimum SoC of each cell (%); \`self.i_discharge\` - the (abs) maximum discharge current of each cell (A);

`self.i_charge` - the (abs) maximum charging current of each cell (A); `self.Power_Output` - the power output of the storage array at each time-step (kW); and `self.SoC` - the SoC of the storage array at each time-step.

ACSE9.StorageModel Algorithm

```
Inputs  eff_round, v_nominal, count, dt,
        q_nominal, SoC_min, i_discharge,
        i_charge, initial SoC
1:      Define array inside
        `ACSE9.StorageModel.StorageArray`
        class.
2:      data in
        PV power generated during time-step
        wind power generated during time-step
        electric load to satisfy over time-step
3:      Calculate, delta = load - generation
4:      if delta > 0 then discharge:
5:        Account for efficiency losses
6:        Calculate required current, I
7:        Calculate current through each branch, i*
*         (Assuming parallel configuration)
8:        Calculate charge consumption, Q
9:        Calculate change in SoC, SoC_delta
10:       if self.SoC + SoC_delta < self.SoC_min
          then not enough charge:
11:         self.Power_Output = 0.0
12:       else then discharge:
13:         if np.abs(i) > self.i_discharge
            then discharge current too large:
14:           self.Power_Output = 0.0
15:         else then discharge:
16:           self.Power_Output = I*self_v_nom
17:           self.SoC += SoC_delta
18:     if delta < 0 then charge:
19:       repeat lines 5-9
20:       if self.SoC == 100
          then cannot accept any more charge:
21:         self.Power_Output = 0.0
22:         skip to line 35
23:       if self.SoC + SoC_delta > 100
          then charge must be curtailed:
25:         calculate charge left to full charge,
            Q = 100 – self.SoC
26:         calculate current required to meet Q,
            I = Q / self.dt
27:         calculate current through each branch, i
28:         if np.abs(i) > self.i_charge
            then charge current too large, curtail:
29:           calculate maximum charge current,
              I = self.i_charge * count
29:           repeat lines 26, 16
30:           self.SoC+= Q/(q_nom*count)
31:         if np.abs(i) <= self.i_charge
            then charge curtailed amount
32:           repeat lines 16, 17
33:       if self.SoC + SoC_delta == 100
          then charge array:
34          repeat lines 28-31
35      end
```