uGrid User Guide

Minigrid Design Toolset

OnePower Africa

Phylicia Cicilio

# Table of Contents - Page Numbers

[Table of Contents - Page Numbers 2](#_Toc10642576)

[Introduction to toolset 7](#_Toc10642577)

[Figure 1: uGrid Toolset Flow Chart 8](#_Toc10642578)

[uGrid 9](#_Toc10642579)

[Quick Start 9](#_Toc10642580)

[Inputs 10](#_Toc10642581)

[Main Inputs: uGrid\_Input.xlsx 10](#_Toc10642582)

[PSO Sheet 10](#_Toc10642583)

[Econ Sheet 10](#_Toc10642584)

[Tech Sheet 11](#_Toc10642585)

[Solar Sheet 12](#_Toc10642586)

[Load Data: LoadKW\_MAK.xls 12](#_Toc10642587)

[Load Forecasting Data: FullYearEnergy.xlsx 12](#_Toc10642588)

[Weather Data: MSU\_TMY.xlsx 12](#_Toc10642589)

[Macro 13](#_Toc10642590)

[Figure 2: Particle Swarm Optimization Procedure 14](#_Toc10642591)

[Technical Functions 16](#_Toc10642592)

[batt\_calcs 16](#_Toc10642593)

[Inputs 16](#_Toc10642594)

[Outputs: 16](#_Toc10642595)

[GenControl 17](#_Toc10642596)

[Inputs 17](#_Toc10642597)

[Outputs: 18](#_Toc10642598)

[fuel\_calcs 18](#_Toc10642599)

[Inputs 18](#_Toc10642600)

[Outputs 18](#_Toc10642601)

[Operation 18](#_Toc10642602)

[Inputs 18](#_Toc10642603)

[Outputs 19](#_Toc10642604)

[PlotPowerFlows 19](#_Toc10642605)

[Inputs 19](#_Toc10642606)

[Outputs 19](#_Toc10642607)

[PlotLoad 20](#_Toc10642608)

[Inputs 21](#_Toc10642609)

[Outputs 21](#_Toc10642610)

[Tech\_total 21](#_Toc10642611)

[Inputs 21](#_Toc10642612)

[Outputs 21](#_Toc10642613)

[Run as Standalone 22](#_Toc10642614)

[Solar Functions 22](#_Toc10642615)

[NRELTheta 22](#_Toc10642616)

[Inputs 22](#_Toc10642617)

[Outputs 22](#_Toc10642618)

[SolarZenith 22](#_Toc10642619)

[Inputs 23](#_Toc10642620)

[Outputs 23](#_Toc10642621)

[GetTheta 23](#_Toc10642622)

[Inputs 23](#_Toc10642623)

[Outputs 23](#_Toc10642624)

[GetGT 23](#_Toc10642625)

[Inputs 23](#_Toc10642626)

[Outputs 23](#_Toc10642627)

[GetPVPower 23](#_Toc10642628)

[Inputs 24](#_Toc10642629)

[Outputs 24](#_Toc10642630)

[SolarTotal 24](#_Toc10642631)

[Inputs 24](#_Toc10642632)

[Outputs 24](#_Toc10642633)

[Run as Standalone 24](#_Toc10642634)

[Economic Functions 25](#_Toc10642635)

[mcashflow 25](#_Toc10642636)

[Inputs 25](#_Toc10642637)

[Outputs 25](#_Toc10642638)

[Econ\_total 26](#_Toc10642639)

[Inputs 26](#_Toc10642640)

[Outputs 26](#_Toc10642641)

[Run as Standalone 26](#_Toc10642642)

[Output 26](#_Toc10642643)

[Optimization Output Sheet 26](#_Toc10642644)

[Other Solution Output Sheet 26](#_Toc10642645)

[Record History Sheet 27](#_Toc10642646)

[uGridNet 28](#_Toc10642647)

[Quick Start 28](#_Toc10642648)

[Inputs 29](#_Toc10642649)

[Main Inputs: uGrid\_Input.xlsx 29](#_Toc10642650)

[Exclusion Locations: MAK\_exclusions.xlsx 29](#_Toc10642651)

[Load Data: LoadkW\_MAK.xlsx 29](#_Toc10642652)

[PDF Image of Village: MAK\_exclusions.pdf 30](#_Toc10642653)

[Functions 30](#_Toc10642654)

[GPStoDistance 30](#_Toc10642655)

[Inputs 30](#_Toc10642656)

[Outputs 30](#_Toc10642657)

[ExclusionMapper 30](#_Toc10642658)

[Inputs 30](#_Toc10642659)

[Output 31](#_Toc10642660)

[DistanceBWindexes 31](#_Toc10642661)

[Inputs 31](#_Toc10642662)

[Output 31](#_Toc10642663)

[Clustering 31](#_Toc10642664)

[Inputs 31](#_Toc10642665)

[Outputs 31](#_Toc10642666)

[FindNonExclusionSpot 31](#_Toc10642667)

[Inputs 31](#_Toc10642668)

[Outputs 32](#_Toc10642669)

[MatchPolesConn 32](#_Toc10642670)

[Inputs 32](#_Toc10642671)

[Output 32](#_Toc10642672)

[PolePlacement 32](#_Toc10642673)

[Inputs 32](#_Toc10642674)

[Outputs 33](#_Toc10642675)

[PenaltiesToCost 33](#_Toc10642676)

[Inputs 33](#_Toc10642677)

[Output 33](#_Toc10642678)

[CollectVillageData 33](#_Toc10642679)

[Inputs 33](#_Toc10642680)

[Outputs 33](#_Toc10642681)

[Plot\_AllPoles\_AllWiring 34](#_Toc10642682)

[Inputs 34](#_Toc10642683)

[Outputs 34](#_Toc10642684)

[Check\_connections 34](#_Toc10642685)

[Inputs 34](#_Toc10642686)

[Output 34](#_Toc10642687)

[ConnectedComponents 34](#_Toc10642688)

[Inputs 34](#_Toc10642689)

[Outputs 35](#_Toc10642690)

[PoleLoads 35](#_Toc10642691)

[Inputs 35](#_Toc10642692)

[Output 35](#_Toc10642693)

[LineLosses 35](#_Toc10642694)

[Inputs 35](#_Toc10642695)

[Output 35](#_Toc10642696)

[WiringAlg 35](#_Toc10642697)

[Inputs 36](#_Toc10642698)

[Outputs 36](#_Toc10642699)

[loadBehindPoles 37](#_Toc10642700)

[Inputs 37](#_Toc10642701)

[Outputs 37](#_Toc10642702)

[POI\_Pole 37](#_Toc10642703)

[Inputs 37](#_Toc10642704)

[Outputs 37](#_Toc10642705)

[Run as Standalone 37](#_Toc10642706)

[Inputs 38](#_Toc10642707)

[Output 38](#_Toc10642708)

[Reliability Cost Output 38](#_Toc10642709)

[Total Cost Output 38](#_Toc10642710)

[MV Wiring Layout Output 38](#_Toc10642711)

[MV Pole Placements Output 39](#_Toc10642712)

[Best Group Wiring Layout Output 39](#_Toc10642713)

[Best Group Pole Placements Output 39](#_Toc10642714)

[Best LV pole and house connections output 39](#_Toc10642715)

[Best LV Pole Placements Output 39](#_Toc10642716)

[Best MV pole and house connections output 39](#_Toc10642717)

[Best Total Costs Output 39](#_Toc10642718)

# 

# Introduction to toolset

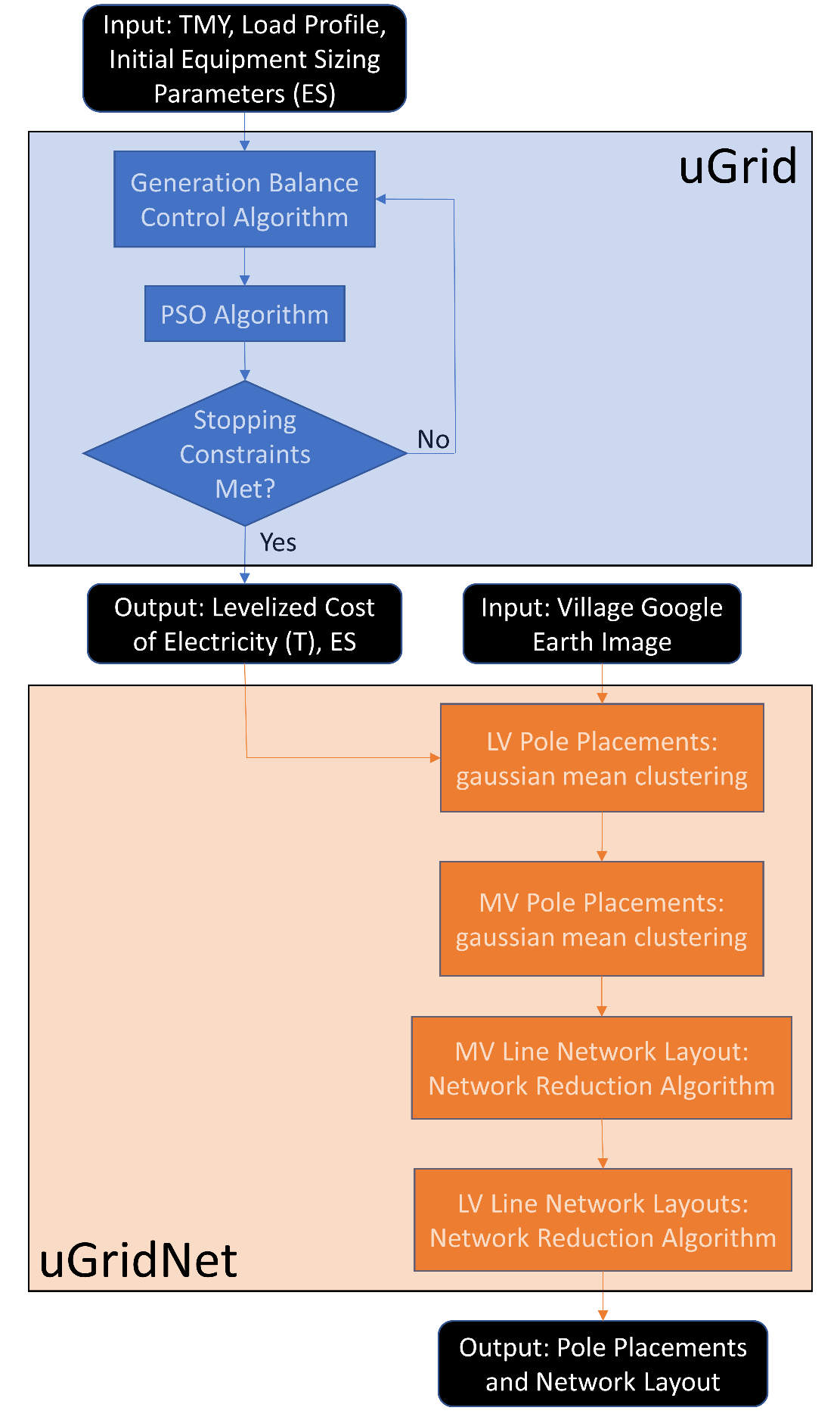
uGrid is an open source toolset written in python to aid the engineering, procurement and construction (EPC) of sustainable and reliable minigrids. This toolset is built by the minigrid developer, OnePower Africa, to meet their development needs including resource sizing and distribution network layout design. The toolset optimizes for a minimal cost of electricity, refered to as the levelized cost of electricity (LCOE). The toolset is composed of two tools, uGrid and uGridNet. This guide is to aid users in running the toolset and adapt the code and inputs for the community of interest.

uGrid performs minigrid resource sizing using particle swarm optimization. There are numerous inputs including weather and load data which are outlined in the Inputs section. The outputs from the program are the equipment sizes (solar PV generation [kW], battery [kWh], and the LCOE [$/kWh]).

uGridNet performs distribution network layout design using gaussian mean clustering and a network reduction algorithm. The main input is a google earth image with drawn exclusion zones (places where poles can’t be placed). The output are plots and locations of where the poles will be located and the wiring layouts.

How each tool within the toolset functions is described in the following chapters. There are areas of improvement which are noted in the descriptions.

The flow of inputs and outputs through the toolset is illustrated below.



##### Figure 1: uGrid Toolset Flow Chart

# uGrid

## Quick Start

The uGrid code has 3 python files: technical equations, economic equations, and the macro which contains the particle swarm algorithm. The program is run from the macro file, and the macro file calls functions in the technical and economic files.

All changes to adapt the code for a specific community are done from the input excel sheet called “uGrid\_Input.xslx”. There are additional spreadsheets for weather and load that need to be within the same folder as the toolset. See the Inputs section for information on what should be changed. Larger changes, such as technology changes, can be done in the code. See the sections on the python files for descriptions of the functions that can be changed for different control algorithms or technology changes.

When the macro code is run the results from each generation from particle swarm optimization are outputted to the command line. An excel spreadsheet is outputted as the specified name in the Input spreadsheet. The output spreadsheet contains the information from each generation, and the global best results from the optimization.

Functions exist in the technical python file to plot power flows. This is a manual process and can be improved by being automated in the code.

To run the uGrid tool, change the inputs in the input spreadsheet and then run the macro python file. The results will be in the outputted excel spreadsheet.

## Inputs

There are 4 excel spreadsheets that need to be in the same folder as the toolset code. These spreadsheets are: Main Inputs: uGrid\_Input.xlsx, Load Data: LoadKW\_MAK.xlsx, Load Forecasting Data: FullYearEnergy.xlsx, Weather Data: MSU\_TMY.xlsx. The details of these spreadsheets are described in the following sections.

### Main Inputs: uGrid\_Input.xlsx

The main inputs to the uGrid program are entered via the uGrid\_Input.xlsx excel spreadsheet. The spreadsheet is separated into sheets with inputs called by each python file. The inputs needed for each sheet are listed and described below.

#### PSO Sheet

|  |  |
| --- | --- |
| **Input** | **Description** |
| maxGen | Maximum number of generations to iteration through |
| numInd | Number of individuals per generation |
| X\_tariff\_multiplier | PSO stopping constraint – see macro section |
| stopLimit | PSO stopping constraint – see macro section |
| convergenceRequirement | PSO stopping constraint – see macro section |
| lowTestLim | PSO stopping constraint – see macro section |
| highTestLim | PSO stopping constraint – see macro section |
| roundDownSize | Equipment sizes are rounded to 0 if below this number |
| C1 | PSO parameter – see macro section |
| C2 | PSO parameter – see macro section |
| CF | PSO parameter – see macro section |
| W | PSO parameter – see macro section |
| VF | PSO parameter – see macro section |
| momentum | PSO parameter – see macro section |
| output\_name | Name of output spreadsheet |

#### Econ Sheet

|  |  |
| --- | --- |
| **Input** | **Description** |
| lifetime | Lifetime of project |
| f\_pv | Factors for distributing maintenance costs in time as a function of capex see Orosz IMechE |
| a\_pv | Factors for distributing maintenance costs in time as a function of capex see Orosz IMechE |
| f | Factors for distributing maintenance costs in time as a function of capex see Orosz IMechE |
| a | Factors for distributing maintenance costs in time as a function of capex see Orosz IMechE |
| Interest\_rate | Interest rate of loan |
| term | Term of loan |
| loanfactor | Loan factor |
| Equity\_debt\_ratio | Equity debt ratio |
| Batt\_lifecycle | Used to estimate years of battery life before replacement is necessary |
| Node\_num | Number of I/O meters \*\*FIX cost is hard input |
| Dist\_km | Estimate for number of poles: 1 pole for every 50m distribution wire. |
| Step\_up\_Trans\_num | Number of step up transformers |
| Pole\_Trans\_num | Number of transmission power poles (6.3kV) |
| Cost\_Dist\_wire | Cost of distribution line [$/km] (220V) |
| Cost\_batt | Cost of battery [$/kWh] |
| Cost\_panel\_per\_kW | Cost of solar panel per kW [$/kW] |
| Cost\_control |  |
| Cost\_charge\_controllers\_per\_kW |  |
| Cost\_Pole | Cost of power pole [$/pole] |
| Cost\_Pole\_Trans | Cost of step down transformer [$/transformer] |
| Cost\_Step\_up\_Trans | Cost of step up transformer [$/transformer] |
| Cost\_Mpesa\_per\_kWLoad | Estimate for merchant services with vodacom [$] |
| Cost\_EPC\_tracker\_per\_kW | Cost of solar tracker per solar panel kW [$/kW] |
| Cist\_EPC\_LPG\_tank | Cost of liquid propane tank [$/tank] |
| Cost\_EPC\_Power\_house | Cost of power house [$/power house] |
| Cost\_EPC\_Labor\_Plant |  |
| Cost\_EPC-Labor\_Dist |  |
| Cost\_Dev\_land |  |
| Cost\_Dev\_EIA |  |
| Cost\_Dev\_connection |  |
| Cost\_Dev\_ICT |  |
| Cost\_Dev\_contingency |  |
| Cost\_Dev\_overhead |  |
| Cost\_taxes | Cost of taxes [$] |
| Tariff\_hillclimber\_multiplier | Tariff is increased by this factor until zero balance is met |
| Cost\_Trans\_wire | Cost of transmission line [$/km] (6.3kV) |
| Cost\_Housing\_Wiring | Connection line cost [$/km] (to from LV to houses) |

#### Tech Sheet

|  |  |
| --- | --- |
| **Input** | **Description** |
| Smart | Binary decision factor to activate load left consideration to generation and balance control algorithm [1/0] |
| Peakload\_buffer | Increases the peak load from the maximum from the load spreadsheet to provide safety buffer of estimated peak load amount |
| Batt\_Charge\_Limit | Percentage of battery charge kept to prevent full discharge and battery free space left to prevent over charging [%] |
| Trans\_losses | Estimated percent of power lost from line losses [%] |

#### Solar Sheet

|  |  |
| --- | --- |
| **Input** | **Description** |
| Year | TMY input sheet |
| Longitude | Community longitude [degrees] |
| Latitude | Community latitude [degrees] |
| Timezone | Time zone GMT (Lesotho is +2) |
| Slope | Slope of solar panels |
| Azimuth | Azimuth of solar panels |
| pg | Ground reflectance, aka albedo [%]. A typical value for grass-covered areas is 20%. Snow-covered areas may have a reflectance as high as 70%. |
| fpv | PV derating factor [%] |
| Alpha\_p | Temperature coefficient of power [%/C]. Average Values: https://www.homerenergy.com/products/pro/docs/3.10/pv\_temperature\_coefficient\_of\_power.html |
| Eff\_mpp | Maximum power point efficiency [%] |
| F\_inv | Inverter efficiency [%] |

### Load Data: LoadKW\_MAK.xls

This spreadsheet consists of 1 column containing the estimated kW power demand for each hour for a year. There is no header.

### Load Forecasting Data: FullYearEnergy.xlsx

This spreadsheet is formatted the same as Load Data spreadsheet, consisting of 1 column containing the estimated kW power demand left for the duration of the night for each hour for a year. There is no header.

The code can be improved by changing the way the load left for the night is forecasted. Ideally, these number will be produced within the code just using the Load Data spreadsheet with no additional Load Forecasting Data spreadsheet needed.

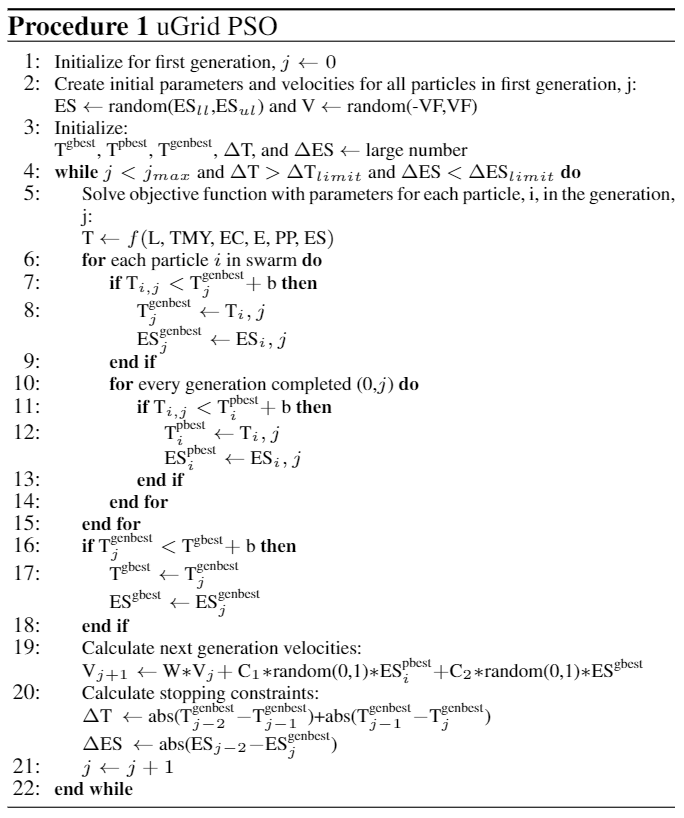
### Weather Data: MSU\_TMY.xlsx

This spreadsheet contains the typical meteorological year (TMY) data for a local location. These data can be obtained from online resources. The necessary TMY data columns are global horizontal irradiance (GHI) with the column heading GHI, ambient temperature with the column heading Tamb, and the hour of the year with the column heading Hour. These data are called by the column headings.

## Macro

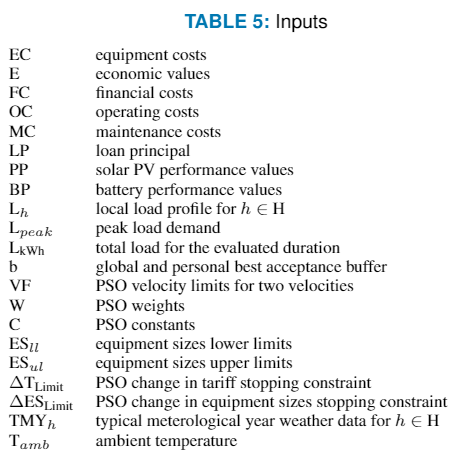
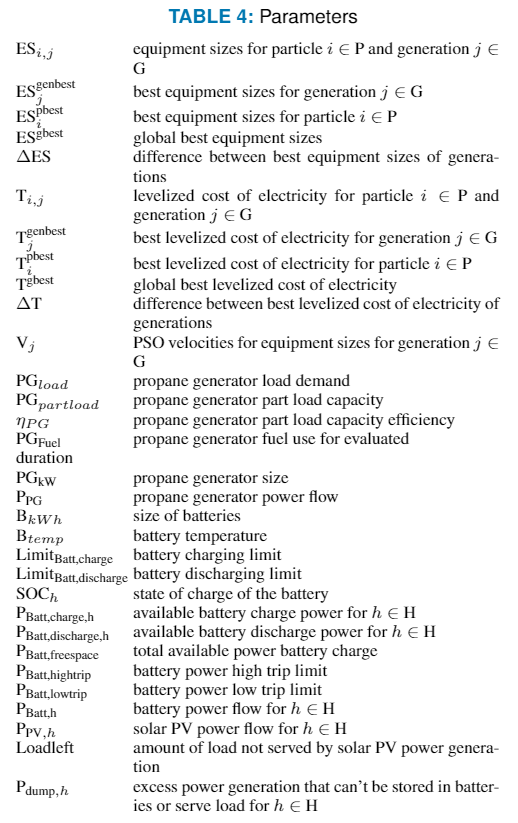
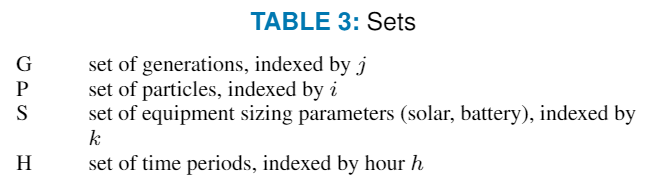
The macro python file runs the uGrid tool code and provides the results from the code. The macro file is all one function, and it runs when the file is run.

The optimization is particle swarm optimization. The algorithm is outlined in the following procedure.



##### Figure 2: Particle Swarm Optimization Procedure

The inputs and parameters outlined in Figure 2 are detailed in the following tables.



## Technical Functions

### batt\_calcs

This function performs the battery performance calculations based on temperature and the limits of the battery. The output is the available charge and discharge amounts and the total freespace in the battery.

#### Inputs

Timestep: the duration of a step in time, currently this is hourly (1hr).

BattkWh: the battery size (kWh)

T\_amb: the ambient temperature (Celsius)

Batt\_SOC\_in: the state of charge of the battery going into the timestep (%)

Limit\_charge: the maximum power charging capacity of the battery at any timestep (kW)

Limit\_discharge: the maximum power discharging capacity of the battery at any timestep (kW)

#### Outputs:

Batt\_discharge: the available power discharging capacity for the timestep (kWh/h)

Batt\_charge: the available power charging capacity for the timestep (kWh/h)

Freespace\_discharge: the total available space in the battery (kWh)

### GenControl

This function performs the generation and load balance to determine the power flows from the battery and propane generator. The battery state of charge and the amount of excess generation to dump is also calculated.

The algorithm for determining the power flows is illustrated in the following figure.

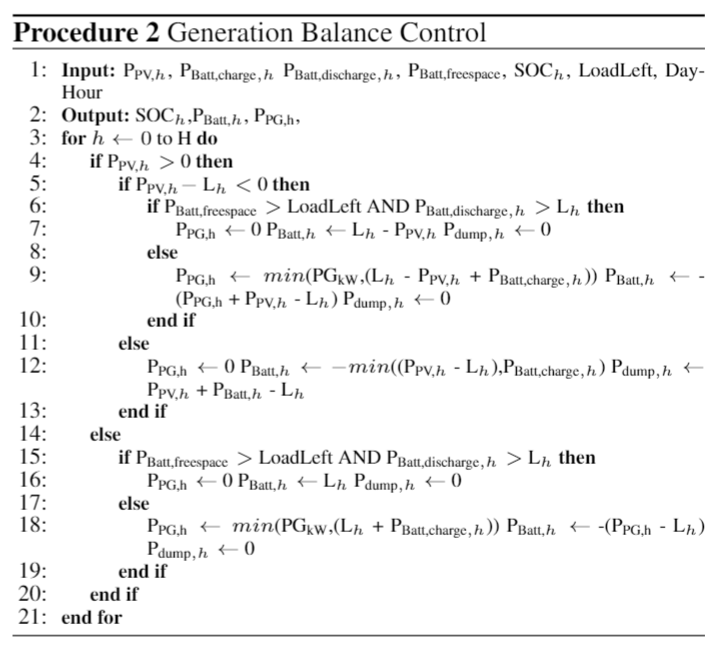


Figure 3: Generation and Load Balance Control Algorithm

#### Inputs

P\_PV: solar PV generation for the timestep (kWh/h)

L: load demand for the timestep (kWh/h)

Batt\_discharge: the available power discharging capacity for the timestep (kWh/h)

Batt\_charge: the available power charging capacity for the timestep (kWh/h)

Freespace\_discharge: the total available space in the battery (kWh)

genPeak: the maximum generating capacity of the propane generator (kW)

LoadLeft: the forecasted amount of load to serve through the coming night (kWh)

Batt\_SOC: the state of charge of the battery entering the timestep (%)

Dayhour: Hour of the day on 24-hour clock (hr)

#### Outputs:

P\_batt: power flow from the battery for the time step. Negative means consuming (charging), positive mean sending power to the grid (discharging). (kWh/h)

P\_gen: power flow from the propane generator for the time step (kWh/h)

Batt\_SOC: the state of charge of the battery leaving the timestep (%)

P\_dump; the amount of power being dumped due to excess solar PV generation (kWh/h)

### fuel\_calcs

This function calculates the amount of propane fuel used by the propane generator for the timestep. The current calculation of fuel is based on an interpolation of the specification of a Onan 25KY propane generator model.

#### Inputs

Genload: the power output of the propane generator for the timestep (kWh/h)

Peakload: the maximum power capacity of the propane generator (kW)

Timestep: the duration of a step in time, currently this is hourly (1hr).

#### Outputs

Fuel\_kW: amount of fuel used by propane generator for the timestep (kW)

Fuel\_kg: amount of fuel used by propane generator for the timestep (kg)

### Operation

This function cycles through the year of data provided to determine the power flows and propane consumption for each timestep in the year by calling all of the previous functions and the solar python file.

#### Inputs

Batt\_Charge\_Limit: Percentage of battery charge kept to prevent full discharge and battery free space left to prevent over charging [%]

Smart: Binary decision factor to activate load left consideration to generation and balance control algorithm [1/0]

PVkW: the size of the solar PV panels (kW)

BattkWh: the size of the batteries (kWh)

Peakload: the estimated maximum load demand (kW)

LoadKW\_MAK: the load demand for each timestep for the year (kWh/h)

FullYearEnergy: the forecasted load demand for during the coming night for the year (kWh/h)

MSU\_TMY: the weather data for each timestep throughout the year

Solar\_Parameters: the parameters from the solar sheet of the uGrid\_Input spreadsheet

Trans\_losses: the estimated percentage of power lost to line losses (%)

#### Outputs

Propane: the amount of propane used throughout the year (gal)

Batt\_SOC; the state of charge of the battery at each timestep throughout the year (%)

LoadkW; the load demand at each timestep throughout the year (kWh/h)

P\_gen; the propane generation at each timestep throughout the year (kWh/h)

P\_PV: the solar PV generation at each timestep throughout the year (kWh/h)

P\_batt: the battery power flow at each timestep throughout the year (kWh/h) (+: discharging, -: charging)

P\_dump: the amount of power dumped at each timestep throughout the year (kWh/h)

Limit\_charge: the maximum power charging capacity of the battery at any timestep (kW)

Limit\_discharge: the maximum power discharging capacity of the battery at any timestep (kW)

Batt\_kWh\_tot: total amount of battery charging throughout the year (kWh)

### PlotPowerFlows

This function creates plots of the power flows and battery state of charge for any specified section of time throughout the year. The time period is specified by the starting time t1,

# and the ending time t2.

#### Inputs

P\_PV: the solar PV generation at each timestep throughout the year (kWh/h)

P\_Batt: the battery power flow at each timestep throughout the year (kWh/h) (+: discharging, -: charging)

P\_PG; the propane generation at each timestep throughout the year (kWh/h)

P\_dump: the amount of power dumped at each timestep throughout the year (kWh/h)

SOC; the state of charge of the battery at each timestep throughout the year (%)

LoadkW: the load demand at each timestep throughout the year (kWh/h)

t1: the starting hour to plot

t2: the ending hour to plot

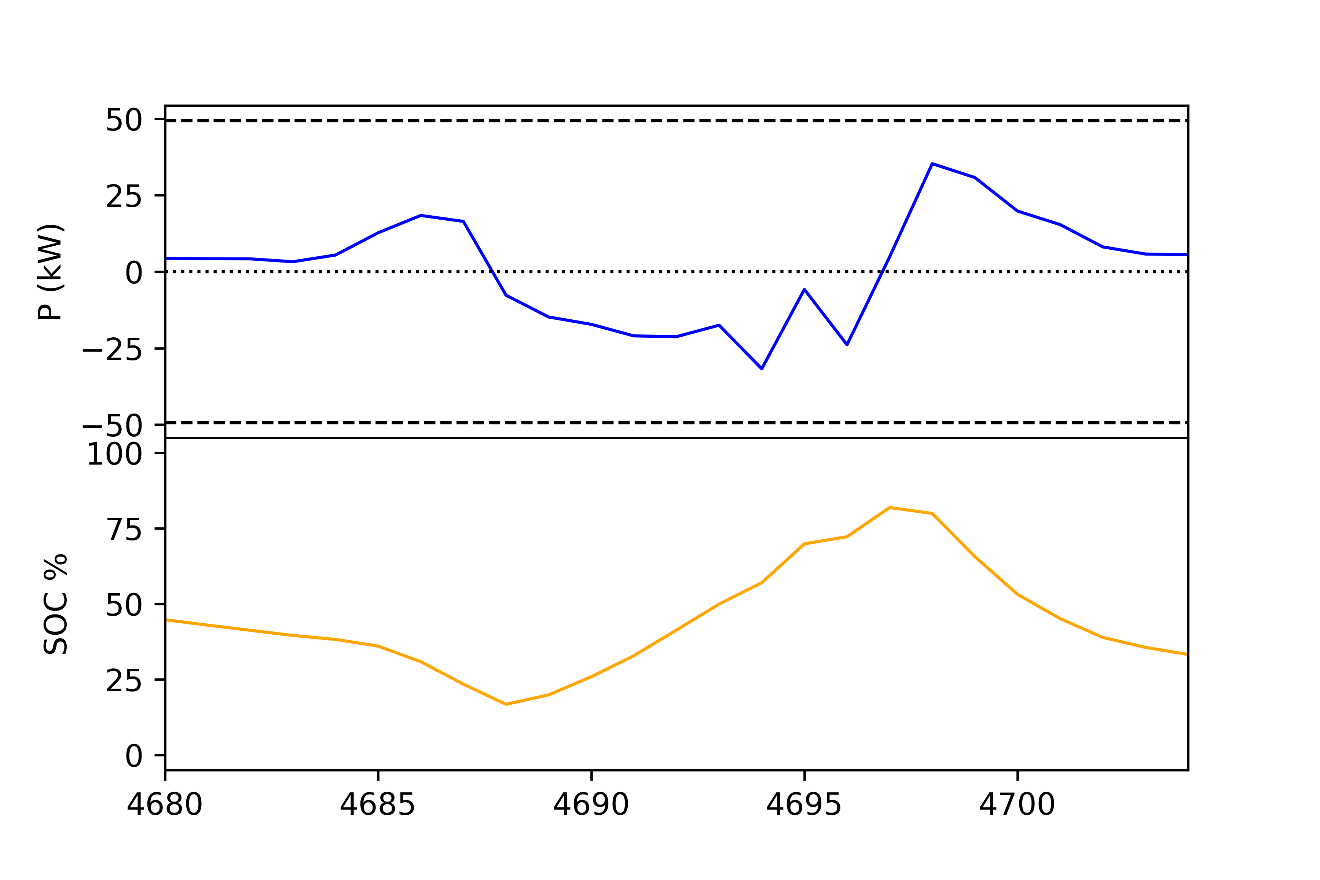
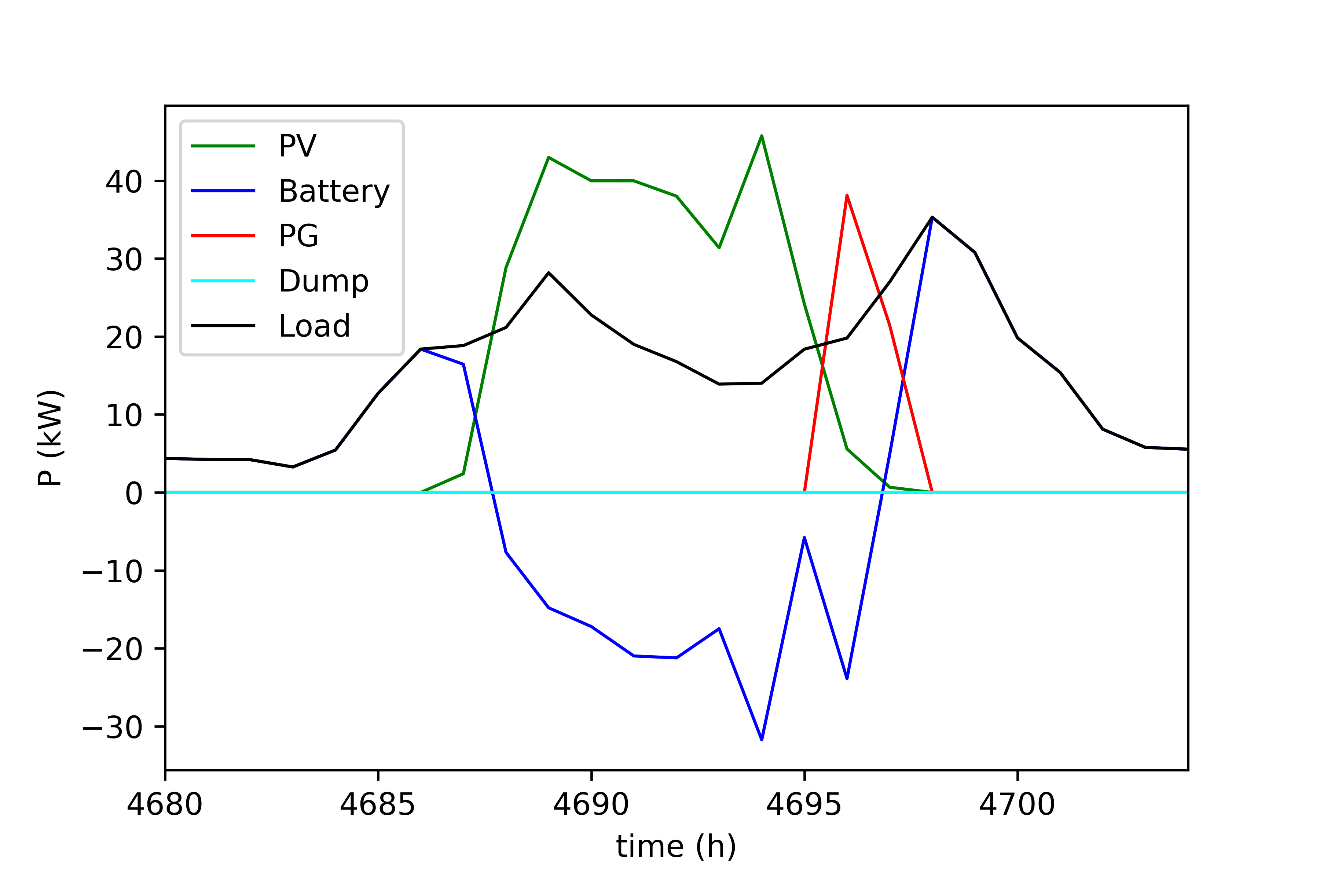
BattkW: the size of the batteries (kWh)

Limit\_charge: the maximum power charging capacity of the battery at any timestep (kW)

Limit\_discharge: the maximum power discharging capacity of the battery at any timestep (kW)

#### Outputs

Below shows example matplotlib plots created by this function.



### PlotLoad

This function plots the load demand from the specified start time, t1, to the end time, t2.

#### Inputs

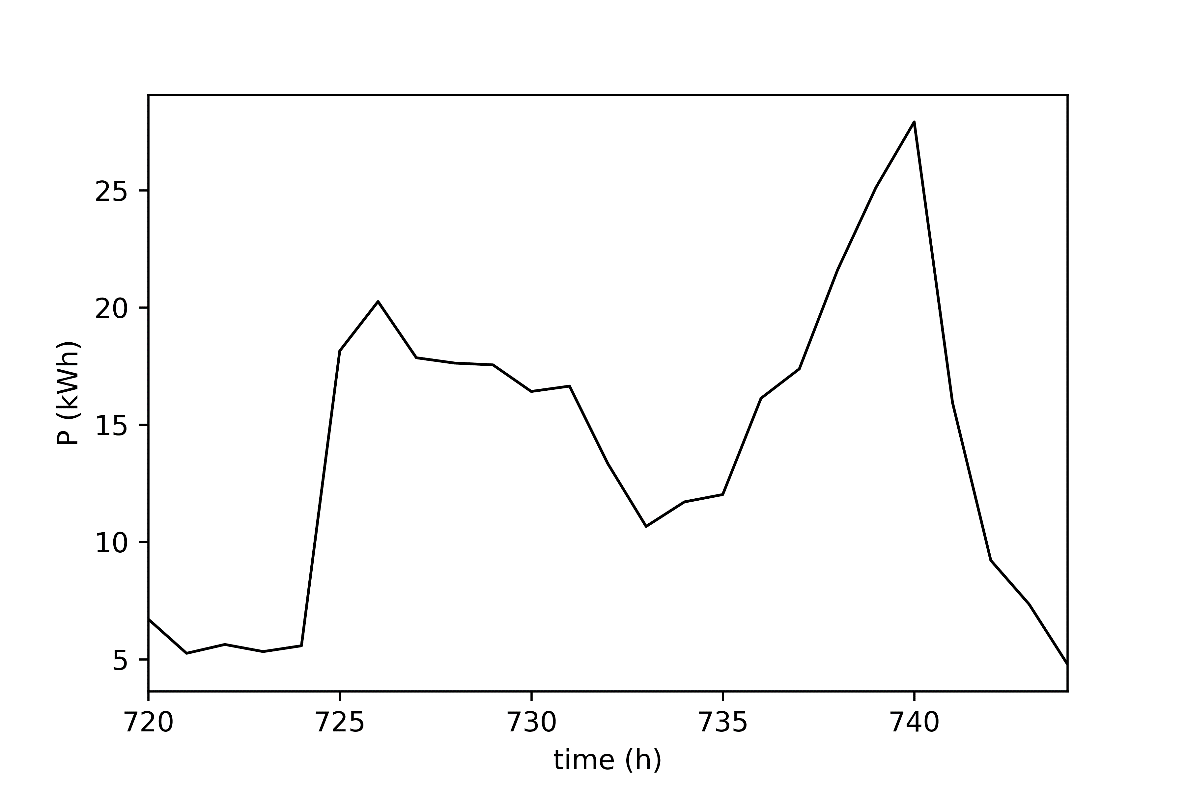
LoadkW: the load demand at each timestep throughout the year (kWh/h)

t1: the starting hour to plot

t2: the ending hour to plot

#### Outputs

Below shows example matplotlib plots created by this function.



### Tech\_total

This function calls the previous functions, excluding the plotting function, to solve for the power flows for the year.

#### Inputs

BattkWh\_Parametric; This is the factor of which times by the peakload provides the battery size in kWh. I suggest this parametric parameter is changed to just the kW size.

PVkW\_Parametric: This is the factor of which times by the peakload provides the solar PV panel size in kW. I suggest this parametric parameter is changed to just the kW size.

#### Outputs

Propane: the amount of propane used throughout the year (gal)

Batt\_SOC; the state of charge of the battery at each timestep throughout the year (%)

LoadkW; the load demand at each timestep throughout the year (kWh/h)

P\_gen; the propane generation at each timestep throughout the year (kWh/h)

P\_PV: the solar PV generation at each timestep throughout the year (kWh/h)

P\_batt: the battery power flow at each timestep throughout the year (kWh/h) (+: discharging, -: charging)

P\_dump: the amount of power dumped at each timestep throughout the year (kWh/h)

Limit\_charge: the maximum power charging capacity of the battery at any timestep (kW)

Limit\_discharge: the maximum power discharging capacity of the battery at any timestep (kW)

Batt\_kWh\_tot: total amount of battery charging throughout the year (kWh)

BattkWh: the size of the batteries (kWh)

loadkWh: the total load demand throughout the year (kWh)

Peakload: the estimated maximum load demand (kW)

### Run as Standalone

This python file can be run as standalone. In order to run as a standalone the BattkWh\_Parametric and PVkW\_Parametric need to be specified. This is also where the plotting functions can be called, and t1 and t2 are specified.

## Solar Functions

The solar functions within the solar calculations python file are all based on how the solar generation calculations are done in Homer.

### NRELTheta

This function calculates the solar hour angle, declination, and day number for the specified hour in the year.

#### Inputs

Year: year of weather data

Hour: hour of the year

Longitude: longitude of the community (degrees)

Latitude: latitude of the community (degrees)

Timezone: time zone of the community in GMT (Lesotho is +2)

#### Outputs

Hrang: the hour angle (degrees)

Declin: the solar declination (radians)

Daynum: the day number

### SolarZenith

This function calculates the solar zenith.

#### Inputs

Declin: the solar declination (radians)

Latitude: latitude of the community (degrees)

Hrang: the hour angle (degrees)

#### Outputs

Zenetr: The solar zenith (degrees)

### GetTheta

This function calculates the solar theta angle.

#### Inputs

Declin: the solar declination (radians)

Latitude: latitude of the community (degrees)

Hrang: the hour angle (degrees)

Slope: the specified slope of the solar panels (degrees)

Azimuth: the specified azimuth of the solar panels (degrees)

#### Outputs

Theta: solar theta (degrees)

### GetGT

This function calculates the global radiation incident on the solar panels.

#### Inputs

G: global horizontal irradiance (GHI)

Slope: the specified slope of the solar panels (degrees)

Pg: ground reflectance, aka albedo [%]

Theta: solar theta (degrees)

Zenetr: The solar zenith (degrees)

Declin: the solar declination (radians)

Latitude: latitude of the community (degrees)

Daynum: the day number

#### Outputs

Gt: global radiation incident on the solar panels

### GetPVPower

This function calculates the solar PV power generation.

#### Inputs

Ypv: the size of the solar PV array (kW)

Fpv: PV derating factor [%]

Alpha\_p: Temperature coefficient of power [%/C]

Tamb: ambient temperature (celsius)

Eff\_mpp: Maximum power point efficiency [%]

F\_inv: Inverter efficiency [%]

#### Outputs

Ppv: solar PV power generation for the specified time step (kW)

### SolarTotal

This function runs all the previously mentioned function to calculate the solar PV power generation for the specified time step.

#### Inputs

MSU\_TMY: weather data

Year: year of weather data

Th\_hour: hour of the year

Longitude: longitude of the community (degrees)

Latitude: latitude of the community (degrees)

Timezone: time zone of the community in GMT (Lesotho is +2)

Azimuth: the specified azimuth of the solar panels (degrees)

Slope: the specified slope of the solar panels (degrees)

Pg: ground reflectance, aka albedo [%]

Ypv: the size of the solar PV array (kW)

Fpv: PV derating factor [%]

Alpha\_p: Temperature coefficient of power [%/C]

Eff\_mpp: Maximum power point efficiency [%]

F\_inv: Inverter efficiency [%]

#### Outputs

Hrang: the hour angle (degrees)

Declin: the solar declination (degrees)

Theta: solar theta (degrees)

Gt: global radiation incident on the solar panels

Ppv: solar PV power generation for the specified time step (kW)

T\_amb: ambient temperature (celsius)

### Run as Standalone

This python file can be run as standalone. In order to run as a standalone all the inputs for the SolarTotal function need to be specified.

## Economic Functions

These functions calculate the tariff, or levelized cost of electricity.

### mcashflow

This function calculates the tariff based on the inputted equipment sizes and minigrid costs and financial parameters.

#### Inputs

Tariff\_hillclimb\_multiplier: Tariff is increased by this factor until zero balance is met

Lifetime: lifetime of the project in years

f\_pv: Factors for distributing maintenance costs in time as a function of capex see Orosz IMechE

a\_pv: Factors for distributing maintenance costs in time as a function of capex see Orosz IMechE

f: Factors for distributing maintenance costs in time as a function of capex see Orosz IMechE

a: Factors for distributing maintenance costs in time as a function of capex see Orosz IMechE

Batt\_life\_yrs: lifespan of the batteries based on the amount of charging (yrs)

Equity\_debt\_ratio

Term: term of loan

loadkWh: the total load demand throughout the year (kWh)

interest\_rate: Interest rate of loan

loanfactor: Loan factor

PVkW: the size of the solar PV panels (kW)

BattkWh: the size of the batteries (kWh)

LEC: levelized cost of electricity starting point for hill climber ($/kWh)

C1\_pv: summation of costs ($)

C1\_LPG: propane generator cost ($)

Cost\_bank: cost of battery bank ($)

Cost\_Propane\_yr: cost of propane usage for a year ($/yr)

#### Outputs

LoanPrincipal: loan principal for the project ($)

Year: array listing the years of the project

Cost: total costs for each year of the project ($/yr)

Revenue: revenue for each year of the project ($/yr)

CashonHand: cash on hand for each year of the project ($/yr)

Balance: balance for each year of the project ($/yr)

M: maintenance cost for each year of the project ($/yr)

O: operating costs for each year of the project ($/yr)

Tariff: tariff or levelized cost of electricity ($/kWh)

### Econ\_total

This function calls in all the necessary economic inputs from the uGrid\_Input.xlsx and calls the mcashflow function to calculate the tariff.

#### Inputs

Propane: the amount of propane used throughout the year (gal)

PVkW: the size of the solar PV panels (kW)

Batt\_kWh\_tot: total amount of battery charging throughout the year (kWh)

Peakload: the estimated maximum load demand (kW)

loadkWh: the total load demand throughout the year (kWh)

#### Outputs

LoanPrincipal: loan principal for the project ($)

Year: array listing the years of the project

Cost: total costs for each year of the project ($/yr)

Revenue: revenue for each year of the project ($/yr)

CashonHand: cash on hand for each year of the project ($/yr)

Balance: balance for each year of the project ($/yr)

M: maintenance cost for each year of the project ($/yr)

O: operating costs for each year of the project ($/yr)

Tariff: tariff or levelized cost of electricity ($/kWh)

Batt\_life\_yrs: lifespan of the batteries based on the amount of charging (yrs)

### Run as Standalone

This python file can be run as standalone. In order to run as a standalone all the inputs for the Econ\_Total function need to be specified.

## Output

The results from each generation is output to the command line. Once the macro has completed running the output excel spreadsheet will be in the file folder. The contents of the output spreadsheet are detailed below by sheet name.

### Optimization Output Sheet

The first column, Column A, is header-less and denotes the generation. Columns B and C show the best equipment sizes in the generation and the resulting propane consumption for the year in gallons, and the resulting tariff in $/kWh.

### Other Solution Output Sheet

Column B contains the estimated total cost of the minigrid for the global best. Column C contains the total time it took to run the program in seconds.

### Record History Sheet

This sheet shows the current global best for each generation. This is not the generation best, but what is set as the global best after completion of each new generation.

# uGridNet

## Quick Start

The uGridNet code goal is to take a kml file which has highlighted polygons of "can't build here" areas and also an inputted generation station GPS location and housing GPS locations, and determine where to place distribution poles and how to connect the distribution network together. This creates a LV 220V and MV 6.3kV network layout, where the MV works as a backbone.

There is one python file called uGrid\_Net\_LVMV\_Reliability.py. The file contains several functions which work together to solve for a distribution network layout including pole placements and wiring. There are 3 excel spreadsheets that need to be in the file folder: main inputs: uGrid\_Input.xlsx, exclusion locations: MAK\_exclusions.xlsx, and load data: LoadkW\_MAK.xlsx. Additionally, the kml image with the exclusion zones needs to be converted to a pdf, this can be completed with free online program.

The uGridNet code use gaussian mean clustering for initial pole placements. The pole placements are then adjusted to ensure they are not placed on exclusion zones. The wiring layout is determined with a wiring layout network reduction algorithm.

Reliability cost benefit analysis consider the loss of revenue from N-1 line loss contingencies is included. It considers if the cost savings from removal of a line outweighs the reliability cost. The reliability cost calculations are covered in the functions described below. The reliability cost benefit can be ignored by setting the probability to zero.

All changes to adapt the code for a specific community are done from the input excel sheets and the inputted images.

There are several output spreadsheets which denote the cost of the system, pole locations, and wiring connections. The output spreadsheets are explained in the output section.

To run the uGrid tool, change the inputs and then run the python file. The results will be in the outputted excel spreadsheets.

Improvement Need: Currently the locations of the poles are outputted as indexes in the image. This needs to be changed to provide GPS locations. This can be done by creating a function that converts the indexes to distances from the edges of the images, and from the distances from the edges the GPS locations can be calculated.

## Inputs

There are 3 excel spreadsheets that need to be in the same folder as the toolset code. These spreadsheets are: main inputs: uGrid\_Input.xlsx, exclusion locations: MAK\_exclusions.xlsx, and load data: LoadkW\_MAK.xlsx. The details of these spreadsheets are described in the following sections. Additionally, the kml image with the exclusion zones needs to be converted to a pdf, this can be completed with free online program.

### Main Inputs: uGrid\_Input.xlsx

This is the same main inputs spreadsheet from the uGrid tool. The spreadsheet labeled Net is used for this tool. The inputs needed for the Net sheet are listed and described below.

|  |  |
| --- | --- |
| **Input** | **Description** |
| reformatScaler | Factor how many pixels are contained in a single index in a program. This effectively adjusts the resolution of the image. |
| exclusionBuffer | The number of indexes from an exclusion zone should be consider also an exclusion zone to provide a buffer. |
| MaxDistancePoleConn | The maximum distance connection poles should be from the houses it connects to. |
| minPoles | The minimum number of poles to test |
| maxPoles | The maximum number of poles to test |
| Range\_limit | The maximum number of indexes a pole could be move to adjust for exclusion zones. |
| Lat\_Generation | Latitude location of the generation station (degrees) |
| Long\_Generation | Longitude location of the generation station (degrees) |
| Cost\_kWh | This is the tariff obtained from the uGrid tool |
| Restoration\_time\_MV | The estimated time to restore a 6.3kV line outage (hours) |
| Prob\_MV | The estimated probability of a 6.3kV line outage (%) |
| Restoration\_time\_LV | The estimated time to restore a 220V line outage (hours) |
| Prob\_LV | The estimated probability of a 220V line outage (%) |

### Exclusion Locations: MAK\_exclusions.xlsx

This spreadsheet is created by converting the kml image with exclusion zones of the village to an excel spreadsheet. This can be done online using free programs.

### Load Data: LoadkW\_MAK.xlsx

This spreadsheet is the same load data spreadsheet used in the uGrid tool.

### PDF Image of Village: MAK\_exclusions.pdf

This is a pdf image of the kml image including exclusions from the village. This can be created from converting the kml to pdf using a free online program.

## Functions

### GPStoDistance

This function calculates the distance between two GPS locations.

#### Inputs

Lat1: Latitude of location 1 (radians)

Lat2: Latitude of location 2 (radians)

Long1: Longitude of location 1 (radians)

Long2: Longitude of location 2 (radians)

#### Outputs

D: distance between location 1 and location 2 (m)

### ExclusionMapper

This function takes the exclusion array, collected from the pdf image, and recasts the exclusion zones into a new array accounting the for the exclusion buffer and reformat scaler. The reformat scaler is used to reduce the dimensions of the exclusion array so the program can run faster. Additionally, it can make sense to reduce the resolution if the original pixel resolution is less than a foot, such a high resolution is unnecessary.

#### Inputs

ExclusionMap\_array: exclusion zone array converted from the pdf image. Exclusions are grey and black, and safe zones are white. Therefore any array cell (containing 3 RBG values) that has anything other than 0 (white) is an exclusion zone.

reformatScaler: Factor how many pixels are contained in a single index in a program. This effectively adjusts the resolution of the image.

exclusionBuffer: The number of indexes from an exclusion zone should be consider also an exclusion zone to provide a buffer.

d\_EW\_between: distance east to west between pixels (m)

d\_NS\_between: distance north to south between pixels (m)

width\_new: number of array indexes the new exclusion zone map will be wide

height\_new: number of array indexes the new exclusion zone map will be tall

#### Output

Indexes: exclusion zone array of ones and zeros accounting for exclusion buffer and reformat scaler

### DistanceBWindexes

This function calculates the distances between array indexes. This works for any array, just the distance between indexes (or “pixels”) needs to be inputted.

#### Inputs

indexesA: array of a set of indexes

indexesB: array of a set of indexes to compare to indexesA

d\_EW\_between: distance east to west between pixels (m)

d\_NS\_between: distance north to south between pixels (m)

#### Output

DistanceAB: Array or single value of the distance between the inputted indexes.

### Clustering

This function clusters the house location to obtain initial placements of the LV (220V) poles. The number of clusters to create is set by the num\_clusters input. The clustering is performed with gaussian mean clustering using the scikitlearn GaussianMixture function.

#### Inputs

Indexes\_conn: array of indexes of the house locations

Num\_clusters: the number of clusters to create out of the house locations

#### Outputs

Y\_: the index is the house and the value is the pole the house is connected to.

Means: the location (indexes) of the mean center of the cluster, where the pole is placed.

### FindNonExclusionSpot

This function determines if the initial pole placement is in an exclusion zone. If the pole is in an exclusion zone, the function tests the area around the initial placement in a circular fashion to find the closest index location to the initial placement that is not an exclusion zone.

#### Inputs

Index\_x\_og: original x index value of pole

Index\_y\_og: original y index value of pole

Index\_excl\_comp: indexes of the exclusion zones

Range\_limit: The maximum number of indexes a pole could be move to adjust for exclusion zones.

Max\_y: the largest y index value in the array

Max\_x: the largest x index value in the array

#### Outputs

The output is two single values representing the x index and y index of the new pole placement.

X: x index location

Y: y index location

### MatchPolesConn

This function creates an array to specify which poles connect to which houses, by which houses are closest to which poles.

#### Inputs

Indexes\_conn: indexes of the houses

Indexes\_poles: indexes of the poles

d\_EW\_between: distance east to west between indexes (m)

d\_NS\_between: distance north to south between indexes (m)

#### Output

ConnPoles: array size (# of connections, 2) where the first column is the pole number which connects to the house and the second column is the distance to that pole.

### PolePlacement

This function runs the previously mentioned functions together to determine the final pole placements.

#### Inputs

reformatScaler: Factor how many pixels are contained in a single index in a program. This effectively adjusts the resolution of the image.

num\_cluster: the number of clusters to create out of the house locations

exclusionBuffer: The number of indexes from an exclusion zone should be consider also an exclusion zone to provide a buffer.

Range\_limit: The maximum number of indexes a pole could be move to adjust for exclusion zones.

Indexes\_conn: indexes of the houses

indexes\_excl: indexes of the exclusion zones

height: number of indexes the exclusion map is tall

width: number of indexes the exclusion map is wide

d\_EW\_between: distance east to west between indexes (m)

d\_NS\_between: distance north to south between indexes (m)

#### Outputs

ConnPoles

Indexes\_poles\_in\_use\_og: the indexes of the poles that are in use. This excludes any poles that were originally placed by eventually determined not to be used in the function.

### PenaltiesToCost

This function calculates the total cost of the network including the equipment costs and reliability cost. The reliability cost can be set to zero.

#### Inputs

Reliability\_cost: the reliability cost of the network

Conn\_wiring: the length of the connection (to houses) wiring

LV\_wiring: the length of the LV (220V) wiring

MV\_wiring: the length of the MV (6.3kV) wiring

Num\_LV\_poles: the number of LV (220V) poles

Num\_MV\_poles: the number of MV (6.3kV) poles

ConnPoles: array size (# of connections, 2) where the first column is the pole number which connects to the house and the second column is the distance to that pole.

#### Output

Total\_cost: total cost of the network

### CollectVillageData

This function collects all of the data of the community needed to determine the network layout.

#### Inputs

There are no inputs to the function. All inputs are called from the excel spreadsheets and pdf image located in the file folder.

#### Outputs

Indexes\_conn: indexes of the houses

indexes\_excl: indexes of the exclusion zones

height: number of indexes the exclusion map is tall

width: number of indexes the exclusion map is wide

d\_EW\_between: distance east to west between indexes (m)

d\_NS\_between: distance north to south between indexes (m)

Long\_exc\_max: the maximum longitude in the exclusions array

Long\_exc\_min: the minimum longitude in the exclusions array

Lat\_exc\_max: the maximum latitude in the exclusions array

Lat\_exc\_min: the minimum latitude in the exclusions array

### Plot\_AllPoles\_AllWiring

This function plots all the wiring (MV, LV, and house connections) together on the same plot.

#### Inputs

POI: the index location of the generation station

OnOff\_MV: an array of ones and zeros noting which poles are connected to each other

Indexes\_pole\_MV: indexes of the MV (6.3kV) poles

OnOff\_groups: groups of arrays of ones and zeros noting which LV group poles are connected to each other

Indexes\_poles\_groups: indexes of the groups LV (220V) poles

Indexes\_conn: indexes of the houses

ConnPoles: array size (# of connections, 2) where the first column is the pole number which connects to the house and the second column is the distance to that pole.

Indexes\_poles\_LV\_all: all indexes of the LV (220V) poles

#### Outputs

There are no returned outputs. The output is a plot of the network including all wiring.

### Check\_connections

This function is used internally to the ConnectedComponents and LineLosses functions. It determines which lines have been visited relevant to those functions.

#### Inputs

i: the starting pole to consider

Visited: input array noting whether each line has been previously visited by the calling function

OnOff: array of ones and zeros noting which poles are connected with wires to each other

#### Output

Visited: updated output visited array

### ConnectedComponents

This function performed the graph theory connected components algorithm. It checks whether all lines have been visited to determine is there are islands in the network.

#### Inputs

OnOff: array of ones and zeros noting which poles are connected with wires to each other

#### Outputs

Conn\_comp: number of islands in the network. There should only be 1 island (one big island = minigrid, more than one island means the minigrid is split into disconnected smaller minigrids)

Visited: output array noting whether each line has been visited by the function

### PoleLoads

This function calculates the load behind each pole.

#### Inputs

ConnPoles: array size (# of connections, 2) where the first column is the pole number which connects to the house and the second column is the distance to that pole.

Num\_poles: number of poles in ConnPoles

#### Output

PoleLoad\_matrix: array noting the load behind each pole. The index is the pole number, the value is the load amount (kW) behind that pole.

### LineLosses

This function calculates the load loss due to the loss of the lines in the network.

#### Inputs

OnOff: array of ones and zeros noting which poles are connected with wires to each other

ConnPoles: array size (# of connections, 2) where the first column is the pole number which connects to the house and the second column is the distance to that pole.

Num\_poles: number of poles in ConnPoles

Long\_exc\_min: the minimum longitude in the exclusions array

Lat\_exc\_min: the minimum latitude in the exclusions array

d\_EW\_between: distance east to west between indexes (m)

d\_NS\_between: distance north to south between indexes (m)

indexes\_poles: indexes of the poles

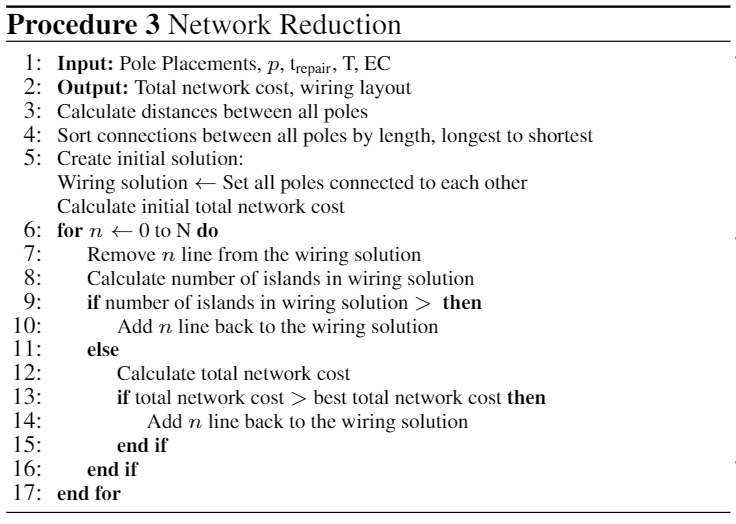
#### Output

Edges: array specifies the amount of load loss due to loss of line, set up in columns: |pole 1|pole 2| load loss|

### WiringAlg

This function determines the lowest cost network layout using a network reduction algorithm by using the previously mentioned functions.

The network reduction algorithm is outlined in the following procedure.



#### Inputs

ConnPoles: array size (# of connections, 2) where the first column is the pole number which connects to the house and the second column is the distance to that pole.

Prob: reliability probability

Cost\_kWh: tariff ($/kWh)

Restoration\_time: estimated time to restore line after line losses

Indexes\_poles: indexes of the poles

Wiring\_cost: cost of wiring

d\_EW\_between: distance east to west between indexes (m)

d\_NS\_between: distance north to south between indexes (m)

standalone: True/False is the program is not being run from the macro python file

#### Outputs

Best\_total\_distance: total wiring distance for best solution (m)

OnOff: array of ones and zeros noting which poles are connected with wires to each other

DistanceBWPoles: square array noting the distance between each pole (m)

Num\_conn\_per\_pole: array noting the number of connections attached to each pole

Best\_Reliability\_Cost; reliability cost for the best solution ($)

Best\_Wire\_Cost: wiring cost for the best solution ($)

Total\_time: time to complete this function

### loadBehindPoles

This function determines the amount of load behind each pole to determine if the connected line is rated to handle that load capacity.

#### Inputs

ConnPoles\_MV: array size (# of MV poles, 2) where the first column is the LV pole number which connects to a MV pole (index of the row) and the second column is the distance between the poles.

ConnPoles\_LV: array size (# of connections, 2) where the first column is the pole number which connects to the house and the second column is the distance to that pole.

Num\_conns: number of houses

MV\_pole\_num: number of MV poles

Load\_per\_conn: the estimated load at each house (kW)

#### Outputs

Max\_load\_behind: value of the maximum load behind any MV (6.3kV) pole

ConnPoles\_LVMV: array size (# of connections) where the first column is the MV pole number which connects to the house (index)

### POI\_Pole

This function determines the index location of the generation station and the closest MV (6.3kV) pole to the generation station.

#### Inputs

Lat\_Generation: latitude location of the generation station (degrees)

Long\_generation: longitude location of the generation station (degrees)

Long\_exc\_min: the minimum longitude in the exclusions array

Lat\_exc\_min: the minimum latitude in the exclusions array

d\_EW\_between: distance east to west between indexes (m)

d\_NS\_between: distance north to south between indexes (m)

indexes\_poles: indexes of the MV poles

#### Outputs

Closest\_pole: the MV pole number which is closest to the generation station

Indexes\_gen: index location of the generation station

### Run as Standalone

This python file is run as a standalone program. To adjust the inputs of the tool for a community the inputs described in the inputs section can be changed. Additional inputs are hard coded in this section and are described below.

This main standalone section calls all previously listed function to determine the best network layout. The outputs are excel spreadsheets whose contents are detailed in the next section.

#### Inputs

Repeats\_MV\_clusters: number of times the MV pole clustering will be repeated to test for a better solution

Repeats\_LV\_clusters: number of times the LV pole clustering will be repeated to test for a better solution

Repeats\_LV\_poles: number of times the LV pole placement determination will be repeated to test for a better solution

Repeats\_MV\_poles: number of times the MV pole placement determination will be repeated to test for a better solution

Repeats\_improved\_solution: number of times the entire (LV and MV) pole placement determination will be repeated to test for a better solution

LV: low voltage wiring rating (V)

MV; medium voltage wiring rating (V)

LV\_kW; power capacity of the LV lines

MV\_kW: power capacity of the MV lines

LV\_safetyfactor: rated percentage of line capacity that is allowed in the tool for a safety measure

LV\_kW\_safety: the power capacity of the LV lines accounting for the safety factor

## Output

The results from the uGridNet tool is output to several excel spreadsheets. The contents of the output spreadsheets are detailed below by spreadsheet name.

These outputs can be improved by combining all these separate spreadsheets in a single spreadsheet with different columns or sheets to contain the different data.

Additionally an output needs to be created that contains the GPS locations of all the poles.

### Reliability Cost Output

This spreadsheet is a single sheet containing a single column of data. Each row is for another iteration containing the reliability cost for that iteration.

### Total Cost Output

This spreadsheet is a single sheet containing a single column of data. Each row is for another iteration containing the total cost for that iteration.

### MV Wiring Layout Output

This spreadsheet is a single sheet containing the OnOff matrix of the MV poles.

### MV Pole Placements Output

This spreadsheet is a single sheet containing the x,y index locations of the MV poles. The first column is the x index, the second column is the y index, the row is the MV pole number.

### Best Group Wiring Layout Output

This spreadsheet is a single sheet containing the OnOff matrix for each of the groups of LV poles. Which group the spreadsheet contains is listed in the name of the spreadsheet. The group number is the MV pole number which the LV pole group connects to.

### Best Group Pole Placements Output

This spreadsheet is a single sheet containing the x,y index locations of the groups of LV poles. Which group the spreadsheet contains is listed in the name of the spreadsheet. The group number is the MV pole number which the LV pole group connects to.

### Best LV pole and house connections output

This spreadsheet is a single sheet containing the ConnPole array of the LV poles. The first column is the LV pole number that connects to the house. The second column contains the distance between the LV pole and the house in meters.

### Best LV Pole Placements Output

This spreadsheet is a single sheet containing the x,y index locations of the LV poles. The first column is the x index, the second column is the y index, the row is the LV pole number.

### Best MV pole and house connections output

This spreadsheet is a single sheet containing the ConnPole array of the MV poles. The first column is the LV pole number that connects to the MV pole which is the index. The second column contains the distance between the LV pole and the MV pole.

### Best Total Costs Output

This spreadsheet is a single sheet containing a single column of data. Each row is for another iteration containing the best total cost for that iteration.