Documentation: Off-grid technology cost competitiveness model

Introduction

This documentation describes the code that is used to evaluate the cost competitiveness between solar plus battery powered mini-grids(MG) and Standalone systems (SAS) powering a rural village. The code sizes for different solar PV plus battery capacity combinations that meet the annual electricity demand for different loads, and optimizes for the least cost system that meets the Loss of Energy Probability (LOEP) over the project lifetime.

Installation

All libraries needed to execute the scripts have been pre-installed and so the code can be executed upon installation of python on your PC. Both scripts have been developed in Python 3.8.8. PyCharm has been used as the Integrated Development Environment to develop the code (see online guide for how to install PyCharm https://www.jetbrains.com/help/pycharm/installation-guide.html) . The code has two scripts:

- **Functions.py** which contains all the functions needed to size the solar plus battery powered systems i.e. MG or SAS
- **Runner.py** which executes the functions from the Functions.py script

To execute the code, download the zip file and save both scripts in the same local directory, then open both scripts on PyCharm.

Input files

The code functions in the same way when sizing MG and the SAS. The main difference is that the MG sizes for the aggregated load, while the SAS sizes for each individual load

The key inputs into the code are:

- csv files
    - Solar irradiation data (GHI_data.csv)
    - Demand pattern data for different loads (load_data.csv)
- Direct inputs into the Runner.py script
    - Technology component specific properties
    - Load setup (village) characteristics
    - Directories where the LCOE for the MG, SAS(individual loads) and the SAS(weighted average) are to be saved

Executing the code

Instructions on how to execute the code:

1. Download the sample input files – GHI_data.csv and load_data.csv.
2. Check whether the delimiter your PC uses in Ms. Excel is “;” or “,” and update the delimiter on the python code accordingly.
3. Update the respective directories in the Runner.py script as shown below:

```
#Define directories
#loads
load_1_dict = {"load_data_directory":r"C:\load_data.csv",
               "MG_results_directory": r'C:\MG\MG' + ".csv",
               "SAS_results_directory": r'C:\SAS\SAS_combined' + ".csv",
               "weighted_results_directory": r'C:\Weighted\weighted' + ".csv"}
#irradiation data
GHI_data_df = pd.read_csv(r"C:\GHI_data.csv", delimiter=';')
```

Figure 1: Sample of directories

4. Execute the code

Description of code in the scripts

Functions.py

A detailed description of each function is described directly in the scripts. The script has 7 functions

1. `obtain_load_profile` – converts load profiles into dataframes needed to size MG (aggregated) and SAS(individual loads)
2. `pv_load` – calculates the PV capacity based on the ambient temperature and the solar irradiation
3. `battery_specs` – generates battery properties for operation during operation and the battery characteristics at the end of life
4. `inverter_specs` – generates inverter specifications which are provided by the user
5. `gen_capacity_sizing` – executes the dispatch algorithm based on the PV and battery capacity combinations to meet the annual electricity demand. Further details on the dispatch algorithm can be found in the SI document of the publication on “The influence of demand patterns on the cost competitiveness between Mini-grids and Standalone Systems”
6. `distribution_sizing` – generates load setup(village) specific characteristics used to size distribution infrastructure
7. `reduced_tech_cost` – calculates the cost of a technology component that has been replaced at its end of life
8. `lcoe_calculation` – calculates the Levelized cost of Electricity (LCOE) for the sized system

Runner.py

The runner script has two functions- one for MG and one for SAS - which rely on functions from the Functions.py script to execute the code.

1. `MG_LCOE_calculator` – iterates through different solar PV plus battery combinations for the *aggregated load* to determine the least cost combination that meets the Loss of Energy Probability(LOEP)
2. `SAS_LCOE_calculator` – iterates through different solar PV plus battery combinations for each *individual load* to determine the least cost combination that meets the Loss of Energy Probability(LOEP)

The code then executes the code for both functions, then saves the files in their respective folders, and finally calculates a weighted SAS LCOE and saves it in a separate folder.

Results

Execution of the code results in three csv files with a format shown below:

MG results (MG.csv):

	PV capacity (kW)	Battery capacity (kWh)	LCOE (USD/kWh)	Loss of Energy Probability(LOEP)	Annual demand (kWh)
Aggregated load					

SAS results (SAS_combined.csv):

	PV capacity (kW)	Battery capacity (kWh)	LCOE (USD/kWh)	Loss of Energy Probability(LOEP)	Annual demand (kWh)
Load 1					
Load 2					
Load 3					

SAS weighted average LCOE (Weighted.csv):

	Weighted SAS LCOE (USD/kWh)
0	

License

For further information on how to execute the code, reach out to Churchill.agutu@gess.ethz.ch or Churchill.agutu@ethalumni.ethz.ch