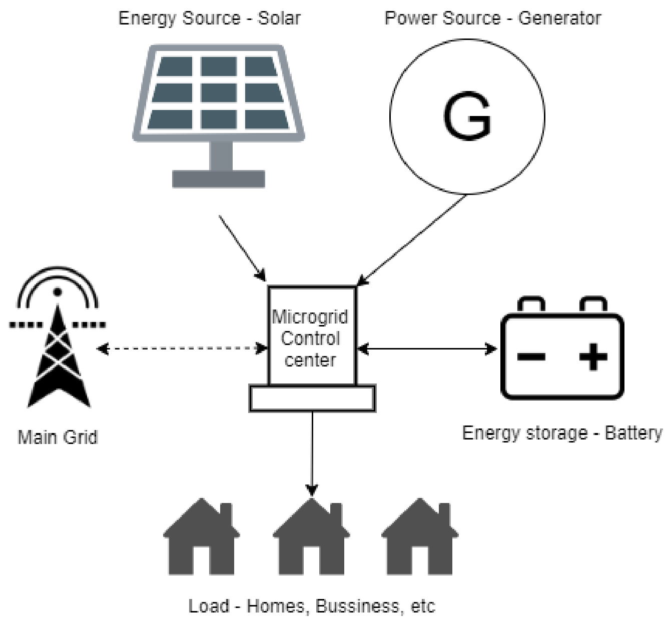
Pymgrid Tool

1. Microgrids

- Demand for the electricity - Load
- Energy source – PV
- Power source – Generator set
- Storage – Battery – charging/discharging power can adjust to balance the system.



Note: grid and genset are controllable.

3 control levels of microgrid

1. Primary – operation of the microgrid itself. OpenAI gym environments target this level of microgrid.
    a. voltage and frequency are controlled.
    b. power sharing
2. Secondary - operation of the microgrid itself.
    a. Voltage and frequency deviation control.
    b. Active and reactive power control.
    c. Grid synchronization
    d. Do power quality and economical operation.
3. Tertiary- coordinate operation of the microgrid and the main grid.
    a. Optimizing cost, efficiency, and other variables.
    b. Coordinate multiple microgrids.
    c. Take decisions for island/interconnect mode.

2. PymGrid

Python library to generate and simulate microgrids. This python package serves as a microgrid virtual environment.

Pymgrid focus on tertiary level operations.

Pros and Cons

Pros	**Cons**
Currently, this is the only open source simulator for large number of microgrids focusing on tertiary control.	Can only change genset and grid in the architecture. (PV and battery are fixed for all the microgrids.)
Possible to generate up to 600 microgrids	
Provide two pre-compute microgrids to use as benchmarks.	
Provide more robust research reproducibility (using benchmarks)	
Easy installation	
Informative Notebooks are available	
Allow RL integration	

2.1 Installation

1. Clone the repo and ***pip install .***
2. Or directly using ***pip install git+https://github.com/Total-RD/pymgrid/***

2. 2 Microgrid Generation

- Possible to create up to 600 microgrids.

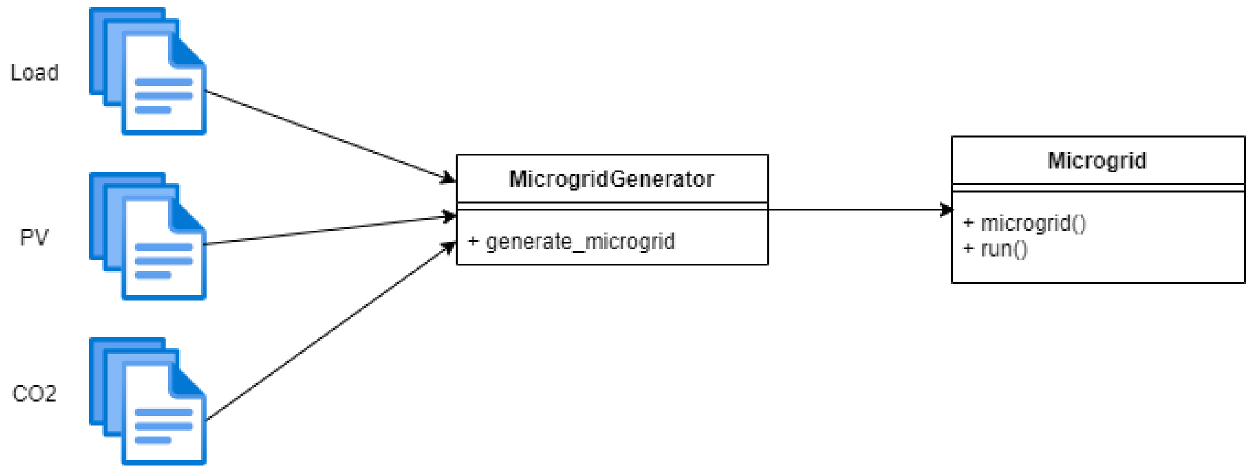
Model Choices

- 5 Load files
- 5 PV files
- 2 Toriffs
- 3 types of grids
- Binary Genset choice

Control parameters – all are optional.

- Number of microgrids to be generated.
- Random seed – this use to generate random numbers to microgrid sizing.

- Timestep – use in time series. (use to generate grid information)
- Path – path to our own data files



2.3 Algorithm – Microgrid generation (return Microgrid object list)

1. **Randomly generate maximum power of the load.**
2. Select load file randomly (1 out of 5)
3. Scale selected load file to the above generated value in step 1 – 2 scaling methods are exist (max or sum)

4. Create microgrid architecture – 3 binary attributes and grid has 3 possible values (grid, no grid and weak grid).
   Architecture - {PV: 1, battery: 1, genset: 0, grid: 1}
   Architecture selection is random.
   a. PV and battery are always present in the architecture (value is always 1)
   b. Present of Generator-set and grid in the microgrid are choose randomly.

   <u>Weak grid scenario</u>
   - Implement a backup generator-set.
   - Electricity tariff is randomly selected if architecture has grid or weak grid.

5. Microgrid Sizing
   a. Calculate PV penetration
   b. Use the calculated PV penetration value to scale the randomly selected PV file. (max or sum – scaling methods)
   c. Battery size (capacity) calculated such a way that battery is capable of delivering power for 3 to 5 hours of mean load)

   d. **Genset size calculated to provide enough power to fulfill the peak load. (take maximum operating loading = 0.9)**
   e. **Calculate grid size such that it is larger than the maximum load (max(load)*2)**
6. Calls microgrid class method.
7. Microgrid class generate the microgrid according to the provided sizing and architecture.

2.4 Microgrid control actions and simulation

We can control the microgrid using control actions.

Function ***run()*** in Microgrid class takes control dictionary as parameter and returns the updated state of the microgrid. (one time step at a time). Can reset the microgrid to initial state using **reset()** method.

Return Object

returns all the parameters that change with the time in microgrid. This is depending on the microgrid architecture.

Example return object fields

```
{
        PV production
        Load
        Battery state of charge
        Battery capacity to charge
        Battery capacity to discharge
        Whether the grid is connected or not
        Co2 intensity of the grid
{
```

Control dictionary

Control dictionary is the way to interact with microgrid class. Allows to pass control commands to the microgrid. Based on the microgrid architecture, these control commands will change.

Example control dictionary

```
{
        ‘battery_charge’:0
        ‘battery_discharge’:1
        ‘grid_import’
        ‘grid_export’
        'PV_consumed'
 }
```

2.5 Benchmark

For benchmarking Pymgrid has pre-defined microgrids (standardize test set)

1. Pymgrid10 – 10 microgrids with same architecture (PV + battery + genset)
2. Pymgrid25 – 25 microgrids with all possible architectures
    a. 4 with only genset
    b. 3 with genset and grid
    c. 9 with only grid
    d. 9 with genset and weak grid

Pymgrid also has pre implemented 4 control algorithms for base line comparison.

1. Rule based control.
2. Model predictive control
3. Q learning
4. Decision tree augmented Q-learning

2.6 Visualization

1. Tool that use in visualization

Use Plotly.offline. python graphing library that can create graphs offline and save them locally.

2.7 CO2 files data

Co2 data is coming from Jacque de Chalendar's grid emissions API. dataset for embodied hourly emissions flows in the US grid.