Optimization of PV-coupled battery systems for combining applications impact of battery technology and location

Generated by Doxygen 1.8.14

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

Namespace Documentation

3.1 Core_LP Namespace Reference

Created on Tue Oct 31 11:11:33 2017 Author Alejandro Pena-Bello alejandro.penabello@unige.ch script used for the paper Optimization of PV-coupled battery systems for combining applications: impact of battery technology and location (Pena-Bello et al 2018 to be published).

Functions

- def **fn_timer** (function)
- def Get_output (instance)
- def Optimize (Capacity, Tech, App_comb, Capacity_tariff, data_input, param, PV_nom)
- def get_cycle_aging (DoD, Technology)
- def aging_day (daily_ESB, SOH, SOC_min, Batt, aux_Cap)
- def single_opt (param, data_input, Capacity_tariff, PV_nom, name)
- def save_results (name, df, Tech, App_comb, Capacity, Cap_arr, SOH, Cycle_aging_factor, P_max, results, cycle_cal_arr, PV_nominal_power, DoD_arr, status)

3.1.1 Detailed Description

Created on Tue Oct 31 11:11:33 2017 Author Alejandro Pena-Bello alejandro.penabello@unige.ch script used for the paper Optimization of PV-coupled battery systems for combining applications: impact of battery technology and location (Pena-Bello et al 2018 to be published).

This script prepares the input for the LP algorithm and get the output in a dataframe, finally it saves the output.

Description

INPUTS

OUTPUTS

TODO

User Interface, including path to save the results and choose countries, load curves, etc.

Requirements

Pandas, numpy, pyomo, pickle, math, sys,glob, time

3.1.2 Function Documentation

3.1.2.1 aging_day()

```
def Core_LP.aging_day (
              daily_ESB,
              SOH,
              SOC_min,
              Batt,
              aux_Cap )
```

```
"
A linear relationship between the capacity loss with the maximum battery
life (years) was chosen. The values of calendric lifetime provide a
reference value for storage degradation to 70% SOH at 20 °C temperature,
when no charge throughput is applied [1].

The temperature is assumed to be controlled and its effect on the battery
aging minimized. As for the cyclic aging, we use a similar approach as the
presented by Magnor et al. assuming different Woehler curves for different
battery technologies [2]. The Woehler curves are provided in the
specifications for some technologies (e.g. NCA), when the curve is not
provided, we use other manufacturer curve, which use the same technology
and adapt it to the referenced number of cycles of the real manufacturer.
The cyclic agingis then the number of cycles per day at the given DoD,
divided by the maximum number of cycles at a given DoD.

[1] H. C. Hesse, R. Martins, P. Musilek, M. Naumann, C. N. Truong, and A.
Jossen, "Economic Optimization of Component Sizing for Residential
Battery Storage Systems," Energies, vol. 10, no. 7, p. 835, Jun. 2017.

[2]D. U. Sauer, P. Merk, M. Ecker, J. B. Gerschler, and D. Magnor, "Concept
of a Battery Aging Model for Lithium-Ion Batteries Considering the Lifetime
Dependency on the Operation Strategy," 24th Eur. Photovolt. Sol. Energy
Conf. 21-25 Sept. 2009 Hambg. Ger., pp. 3128-3134, Nov. 2009.

cycle_cal indicates which aging dominates in the day if 1 cycle is dominating.
Parameters
----------
daily_ESB : array
SOH : float
SOC_min : float
Batt : class
aux_Cap : float

Returns
-------
SOC_max : float
aux_Cap : float
SOH : float
Cycle_aging_factor : float
cycle_cal : int
DoD : float
```

3.1.2.2 get_cycle_aging()

```
def Core_LP.get_cycle_aging (
              DoD,
              Technology )
```

```
The cycle aging factors are defined for each technology according
to the DoD using an exponential function close to the Woehler curve.
Parameters
----------
DoD : float
Technology : string

Returns
-------
Cycle_aging_factor : float
```

3.1.2.3 Get_output()

```
def Core_LP.Get_output (
              instance )
```

```
Gets the model output and transforms it into a pandas dataframe
with the desired names.
Parameters
----------
instance : instance of pyomo

Returns
-------
df : DataFrame
P_max_ : vector of daily maximum power
```

3.1.2.4 Optimize()

```
def Core_LP.Optimize (
              Capacity,
              Tech,
              App_comb,
              Capacity_tariff,
              data_input,
              param,
              PV_nom )
```

```
This function calls the LP and controls the aging. The aging is then
calculated in daily basis and the capacity updated. When the battery
reaches the EoL the loop breaks. 'days' allows to optimize multiple days at once.

Parameters
----------
Capacity : float
Tech : string
App_comb : array
```

```
Capacity_tariff : float
data_input: DataFrame
param: dict
PV_nom: float

Returns
-------
df : DataFrame
aux_Cap_arr : array
SOH_arr : array
Cycle_aging_factor : array
P_max_arr : array
results_arr : array
cycle_cal_arr : array
DoD_arr : array
```

3.1.2.5 save_results()

```
def Core_LP.save_results (
              name,
              df,
              Tech,
              App_comb,
              Capacity,
              Cap_arr,
              SOH,
              Cycle_aging_factor,
              P_max,
              results,
              cycle_cal_arr,
              PV_nominal_power,
              DoD_arr,
              status )


Save the results in pickle format using the corresponding timezone.
The file is saved in the same directory the program is saved.
The name is structured as follows: df_name_tech_App_Cap.pickle
Parameters
----------
name : string
df : DataFrame
Tech : string
App_comb : array
Capacity : float
Cap_arr : array
SOH : float
Cycle_aging_factor : float
P_max :float
results : array
cycle_cal_arr : array
PV_nominal_power : float
DoD_arr : array

Returns
-------
bool
    True if successful, False otherwise.
```

3.1.2.6 single_opt()

```
def Core_LP.single_opt (
              param,
              data_input,
              Capacity_tariff,
              PV_nom,
              name )

"
Iterates over capacities, technologies and applications and calls the module to save the results.
Parameters
----------
param: dict
data_input: DataFrame
Capacity_tariff: float
PV_nom: float
name: string

Returns
-------
df : DataFrame
Cap_arr : array
SOH : float
Cycle_aging_factor : float
P_max : float
results : float
cycle_cal_arr :array
```

3.2 LP Namespace Reference

Created on Wed Nov 1 15:44:25 2017 Author Alejandro Pena-Bello alejandro.penabello@unige.ch In this script the optimization is set up.

Functions

- def **Concrete_model** (Data)
- def **Balance_Batt_rule** (m, i)
- def **E_char_rule** (m, i)
- def **E_dis_rule** (m, i)
- def **Grid_cons_rule** (m, i)
- def **Balance_PV_rule** (m, i)
- def **Sold_rule** (m, i)
- def **Balance_load_rule** (m, i)
- def **def_storage_state_rule** (m, t)
- def **Conv_losses_rule** (m, i)
- def **Inv_losses_PV_rule** (m, i)
- def **Inv_losses_Batt_rule** (m, i)
- def **Inv_losses_Grid_rule** (m, i)
- def **Inv_losses_rule** (m, i)
- def **Batt_losses_rule** (m, i)
- def **SOC_rule** (m, i)
- def **Batt_max_char_rule** (m, i)
- def **Batt_max_dis_rule** (m, i)
- def **Inverter_rule** (m, i)
- def **Converter_rule** (m, i)
- def **Inverter_grid_rule** (m, i)
- def **P_max_rule** (m, i)
- def **Curtailment_rule** (m, i)
- def **PVSC_rule** (m, i)
- def **Obj_fcn** (m)

3.2.1 Detailed Description

Created on Wed Nov 1 15:44:25 2017 Author Alejandro Pena-Bello alejandro.penabello@unige.ch In this script the optimization is set up.

This LP algorithm allows the user to optimize the daily electricity bill The system here presented is a DC-coupled system with a charge controller and a bi-directional inverter as presented in [1] It includes efficiency losses. It includes 4 applications and their combination with PVSC as base app. avoidance of PV curtailment, Demand peak shaving and demand load shifting. Demand peak shaving is done in the same basis than the optimization, i.e. if it is a daily optimization, the peak shaving is done every day, if it is yearly, then the peak of the year is shaved and so on. According to the Data it can be expanded to one month or n years optimization An Integrated Inverter is used, in this script the Converter and inverter efficiency are the same and are an input from the user. The delta_t allows the user to set the time step delta_t=fraction of hour, e.g. delta_t=0.25 is a 15 min time step [1] Installed Cost Benchmarks and Deployment Barriers for Residential Solar+ Photovoltaics with Energy Storage: Q1 2016 Kristen Ardani,Eric O'Shaughnessy,Ran Fu,Chris McClurg,Joshua Huneycutt, and Robert Margolis

| PV |—>| MPPT+Ch.Ctrl|–>|Bi-Inv|--------—|Grid| —- ----------— ----— | -—- | | | | ---------— | | -—- | |Load|

| **Batt** |

3.3 main_paper Namespace Reference

Created on Wed Feb 28 09:47:22 2018 Author Alejandro Pena-Bello alejandro.penabello@unige.ch Main script used for the paper Optimization of PV-coupled battery systems for combining applications: impact of battery technology and location (Pena-Bello et al 2018 to be published).

3.3.1 Detailed Description

Created on Wed Feb 28 09:47:22 2018 Author Alejandro Pena-Bello alejandro.penabello@unige.ch Main script used for the paper Optimization of PV-coupled battery systems for combining applications: impact of battery technology and location (Pena-Bello et al 2018 to be published).

The study focuses on residential PV-coupled battery systems. We study the different applications which residential batteries can perform from a consumer perspective. Applications such as avoidance of PV curtailment, demand load-shifting and demand peak shaving are considered along with the base application, PV self-consumption. Moreover, six different battery technologies currently available in the market are considered as well as three sizes (3 kWh, 7 kWh and 14 kWh). We analyze the impact of the type of demand profile and type of tariff structure by comparing results across dwellings in Switzerland and in the U.S. The battery schedule is optimized for every day (i.e. 24 h optimization framework), we assume perfect day-ahead forecast of the electricity demand load and solar PV generation in order to determine the maximum economic potential regardless of the forecast strategy used. Aging was treated as an exogenous parameter, calculated on daily basis and was not subject of optimization. Data with 15-minute temporal resolution were used for simulations. The model objective function have two components, the energy-based and the power-based component, as the tariff structure depends on the applications considered, a boolean parameter activate the power-based factor of the bill when is necessary. Every optimization was run for one year and then the results were linearly-extrapolated to reach the battery end of life. Therefore, the analysis is done with same prices for all years across battery lifetime. We assume 30% of capacity depletion as the end of life. The script works in Linux and Windows This script works was tested with pyomo version 5.4.3

INPUTS

OUTPUTS

TODO

User Interface, including path to save the results and choose countries, load curves, etc. Simplify by merging select_data and load_data and probably load_param.

Requirements

Pandas, numpy, sys, glob, os, csv, pickle, functools, argparse, itertools, time, math, pyomo and multiprocessing

3.4 paper_classes_2 Namespace Reference

Created on Mon Jul 17 16:43:49 2017 Author Alejandro Pena-Bello alejandro.penabello@unige.ch Battery class definition Based on Parra & Patel, 2016 Prices from ETH, UNIGE, PSI change rate from EUR to USD 1.18 as of August 2017 We can include years of usage as a variable to modify efficiency and include ageing Price in USD/kWh in Pena-Bello et al.

3.4.1 Detailed Description

Created on Mon Jul 17 16:43:49 2017 Author Alejandro Pena-Bello alejandro.penabello@unige.ch Battery class definition Based on Parra & Patel, 2016 Prices from ETH, UNIGE, PSI change rate from EUR to USD 1.18 as of August 2017 We can include years of usage as a variable to modify efficiency and include ageing Price in USD/kWh in Pena-Bello et al.

2017 it changes according to the techno.

Chapter 4

Class Documentation

4.1 classes_p.Battery Class Reference

Public Member Functions

- def **__init__** (self, Capacity, choicetype, kwargs)

Public Attributes

- **Technology**
- **Capacity**

Static Public Attributes

- **Battery_future_Price**
- dictionary **defaults**

4.1.1 Member Data Documentation

4.1.1.1 defaults

```
dictionary classes_p.Battery.defaults  [static]
```

Initial value:

```
=  {'Efficiency': 0.925,
                'P_max_dis': -1*Capacity,
                'P_max_char': 1*Capacity,
                'SOC_max': 1*Capacity,
                'SOC_min': 0*Capacity,
                'Price_battery': 645*Capacity,
                'Battery_cal_life': 20,
                'Battery_cycle_life': 8000,
                'PCS_costs': 286.65*EUR_USD,
                'BOP_costs': 0,
                'OandM_costs': 0,
                'Fix_costs': 0,
                'EoL': 0.7}
```

The documentation for this class was generated from the following file:

- classes_p.py

4.2 classes_p.fullprint Class Reference

Public Member Functions

- def **__init__** (self, kwargs)
- def **__enter__** (self)
- def **__exit__** (self, type, value, traceback)

Public Attributes

- **opt**

The documentation for this class was generated from the following file:

- classes_p.py

4.3 classes_p.Hardware_Prices Class Reference

Public Member Functions

- def **__init__** (self, Inverter_power)

Public Attributes

- **Price_PV**
- **Price_inverter**
- **PV_cal_life**
- **Inverter_cal_life**
- **Interest_rate**

The documentation for this class was generated from the following file:

- classes_p.py