

---

# OMEGAAlpes Documentation

*Release 0.0.1*

**B. DELINCHANT, S. HODENCQ, Y. MARECHAL, L. MORRIET, C. PA**

**Mar 23, 2020**



---

## Contents

---

<b>1</b>	<b>Introduction to OMEGAlpes</b>	<b>1</b>
<b>2</b>	<b>Contents</b>	<b>3</b>
<b>3</b>	<b>Acknowledgments</b>	<b>49</b>
	<b>Python Module Index</b>	<b>51</b>
	<b>Index</b>	<b>53</b>



# CHAPTER 1

---

## Introduction to OMEGAAlpes

---

OMEGAAlpes stands for Generation of Optimization Models As Linear Programming for Energy Systems. It aims to be an energy systems modelling tool for linear optimisation (LP, MILP). It is currently based on the LP modeler PuLP.

It is an Open Source project located on GitLab at <https://gricad-gitlab.univ-grenoble-alpes.fr/omegalpes/omegalpes>



## 2.1 OMEGAlpes Installation

- *Python 3.6.0*
- *pip install omegalpes*

### 2.1.1 Python 3.6.0

Please use Python 3.6.0 for the project interpreter. <https://www.python.org/downloads/release/python-360/>

### 2.1.2 pip install omegalpes

Please install OMEGAlpes Lib with pip using the command prompt

**If you are admin on Windows or working on a virtual environment** `pip install omegalpes`

**If you want a local installation or you are not admin** `pip install --user omegalpes`

**If you are admin on Linux:** `sudo pip install omegalpes`

Then, you can download (or clone) the OMEGAlpes Examples folder (repository) at : <https://gricad-gitlab.univ-grenoble-alpes.fr/omegalpes/omegalpes-examples>

Launch the examples (with Pycharm for instance) to understand how the OMEGAlpes Lib works. Remember that the examples are detailed at : <https://omegalpes-examples.readthedocs.io/en/latest/>

Enjoy your time using OMEGAlpes !

## 2.2 Other installation requirements

Please install the following packages to use OMEGAAlpes if not already installed:

- *PuLP*  $\geq 1.6.8$
- *Matplotlib*  $\geq 2.2.2, <3.1$
- *Numpy*  $\geq 1.14.2, <1.16$
- *Pandas*  $\geq 0.22.0, <0.24$

### 2.2.1 PuLP $\geq 1.6.8$

PuLP is an LP modeler written in python. PuLP can generate MPS or LP files and call GLPK, COIN CLP/CBC, CPLEX, and GUROBI to solve linear problems. <https://github.com/coin-or/pulp>

### 2.2.2 Matplotlib $\geq 2.2.2, <3.1$

Matplotlib is a Python 2D plotting library. <https://matplotlib.org/>

### 2.2.3 Numpy $\geq 1.14.2, <1.16$

NumPy is the fundamental package needed for scientific computing with Python. <https://github.com/numpy/numpy>

### 2.2.4 Pandas $\geq 0.22.0, <0.24$

pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. <https://pandas.pydata.org/pandas-docs/version/0.23.1/index.html>

*Help for library installation:*

Command lover:

```
pip install <library_name>==version
```

If required, the command to upgrade the library is

```
pip install --upgrade <library_name>
```

Pycharm lover:

Install automatically the library using pip with Pycharm on “File”, “settings...”, “Project Interpreter”, “+”, and choosing the required library

## 2.3 Install OMEGAAlpes as a developer

### 2.3.1 Installation as a developer and local branch creation

1. Create a new folder in the suitable path, name it as you wish for example : OMEGAAlpes



## 2. Clone the OMEGAAlpes library repository

— Command lover:

```
git clone https://gricad-gitlab.univ-grenoble-alpes.fr/omegalpes/omegalpes.git
```

— Pycharm lover:

Open Pycharm On the Pycharm window, click on “Check out from version control” then choose “Git”. A “clone repository” window open. Copy the following link into the URL corresponding area:

```
https://gricad-gitlab.univ-grenoble-alpes.fr/omegalpes/omegalpes.git
```

Copy the path of the new folder created just before. Test if the connection to the git works and if it works click on “Clone”. Once OMEGAAlpes is cloned, you must be able to see the full OMEGAAlpes library on Pycharm or on another development environment.

If the connection does not work and if you are working with local protected network, please try again with the wifi.

## 3. First, choose or change your project interpreter

— Pycharm lover:

Click on the yellow warning link or go to “File”, “settings...”, “Project Interpreter”

You can: - either select the “Python 3.6” project interpreter but you may change the version of some library that you could use for another application - either create a virtual environment in order to avoid this problem (recommended). Click on the star wheel near the project interpreter box. Click on “add...”. Select “New environment” if it not selected. The location is pre-filled, if not fill it with the path of the folder as folder\_path/venv Select “Python 3.6” as your base interpreter Then click on “Ok”

—

4. You can install the library on developing mode using the following command in command prompt once your are located it on the former folder. If you are calling OMEGAAlpes library in another project, the following command enables you to refer to the OMEGAAlpes library you are developing

```
python setup.py develop
```

## 5. If it is not already done, install the library requirements.

— Command lover:

```
pip install <library_name>
```

If required, the command to upgrade the library is :

```
pip install --upgrade <library_name>
```

— Pycharm lover:

You should still have a yellow warning. You can: - install automatically the libraries clicking on the yellow bar - install automatically the library using pip with Pycharm on “File”, “settings...”, “Project Interpreter”, “+”, and choosing the required library as indicated in the Library Installation Requirements part

## 6. Finally, you can create your own local development branch.

— Command lover:

```
git branch <branch_name>
```

— Pycharm lover:

By default you are on a local branch named master. Click on “Git: master” located on the bottom write of Pycharm Select “+ New Branch” Name the branch as you convenience for instance “dev\_your\_name”

7. Do not forget to “rebase” regularly to update your version of the library.

— Command lover:

`git rebase origin`

— Pycharm lover:

To do so, click on your branch name on the bottom write of the Pycharm window select “Origin/master” and click on “Rebase current onto selected”

If you want to have access to other examples and study cases, please have a look to the user’s installation.

Enjoy your time developing OMEGAAlpes!

## 2.4 OMEGAAlpes structure

The models are bases on **general**, **energy** and **actor** classes. The structure of the library is described on the following class diagram:

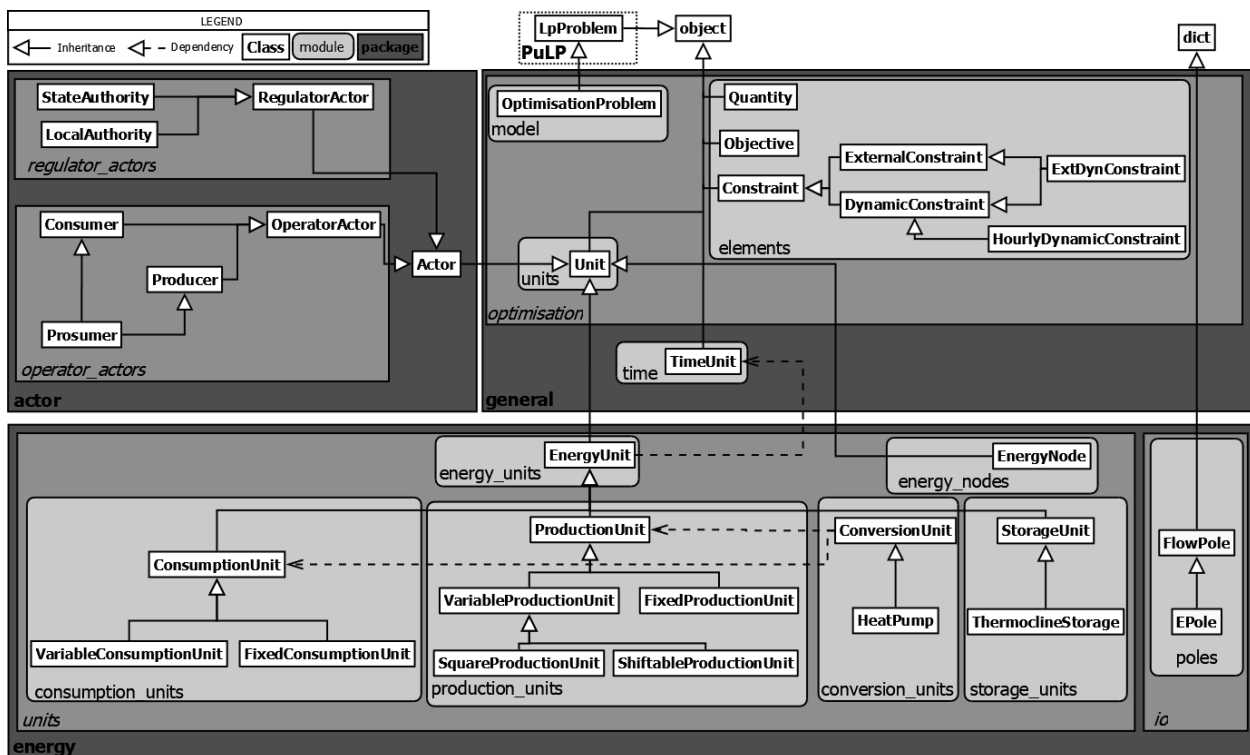


Fig. 1: Figure: OMEGAAlpes class diagram

The energy units are detailed in the energy package

### 2.4.1 energy package

The energy package gathers the energy units, poles and nodes modules:

- *Energy\_units module*
- *Consumption\_units module*
- *Production\_units module*
- *Conversion\_units module*
- *Storage\_units module*
- *Energy\_nodes module*
- *Poles module*

The energy units inherit from the *EnergyUnit* object which itself inherit from the *Unit* object.

#### Energy\_units module

\*\* This module defines the energy units of OMEGALPES. The production, consumption and storage unit will inherit from it. \*\*

The energy\_units module defines the basic attributes and methods of an energy unit in OMEGALPES.

**It includes the following attributes and quantities:**

- *p* : instantaneous power of the energy unit (kW)
- *p\_min* : minimal power (kW)
- *p\_max* : maximal power (kW)
- *e\_tot* : total energy during the time period (kWh)
- *e\_min* : minimal energy of the unit (kWh)
- *e\_max* : maximal energy of the unit (kWh)
- *u* : binary describing if the unit is operating or not at *t* (delivering or consuming P)

Copyright 2018 G2Elab / MAGE

Licensed under the Apache License, Version 2.0 (the “License”); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

```
class omegalpes.energy.units.energy_units.EnergyUnit (time, name,
                                                    flow_direction='in',
                                                    p=None, p_min=-10000.0,
                                                    p_max=10000.0, e_min=-
                                                    1000000.0, e_max=1000000.0,
                                                    starting_cost=None,
                                                    operating_cost=None,
                                                    min_time_on=None,
                                                    min_time_off=None,
                                                    max_ramp_up=None,
                                                    max_ramp_down=None,
                                                    co2_out=None,          avail-
                                                    ability_hours=None,      en-
                                                    ergy_type=None,          opera-
                                                    tor=None)
```

Bases: *omegalpes.general.optimisation.units.Unit*

**\*\* Description \*\***

**Module dedicated to the parent class (EnergyUnit) of :**

- production units
- consumption units
- storage units

**add\_availability** (*av\_hours*)

**add\_co2\_emissions** (*co2\_out*)

**add\_max\_ramp\_down** (*max\_ramp\_down*)

**add\_max\_ramp\_up** (*max\_ramp\_up*)

**add\_min\_time\_off** (*min\_time\_off*)

**add\_min\_time\_on** (*min\_time\_on*)

**add\_operating\_cost** (*operating\_cost*)

**add\_starting\_cost** (*start\_cost*)

**minimize\_CO2\_emissions** (*weight=1*)

Parameters **weight** – Weight coefficient for the objective

Returns

**minimize\_costs** (*weight=1*)

Parameters **weight** – Weight coefficient for the objective

**minimize\_energy** (*weight=1*)

Parameters **weight** – Weight coefficient for the objective

**minimize\_operating\_cost** (*weight=1*)

Parameters **weight** – Weight coefficient for the objective

**minimize\_starting\_cost** (*weight=1*)

Parameters **weight** – Weight coefficient for the objective

**minimize\_time\_of\_use** (*weight=1*)

Parameters **weight** – Weight coefficient for the objective

```
set_energy_limits_on_time_period(e_min=0, e_max=None, start='YYYY-MM-DD
HH:MM:SS', end='YYYY-MM-DD HH:MM:SS',
period_index=None)
```

#### Parameters

- **e\_min** – Minimal energy set during the time period (int or float)
- **e\_max** – Maximal energy set during the time period (int or float)
- **start** – Date of start of the time period YYYY-MM-DD HH:MM:SS ( str)
- **end** – Date of end of the time period YYYY-MM-DD HH:MM:SS (str)

```
class omegalpes.energy.units.energy_units.FixedEnergyUnit (time, name: str, p: list,
                                                            flow_direction='in',
                                                            starting_cost=None,
                                                            operating_cost=None,
                                                            co2_out=None, en-
                                                            ergy_type=None,
                                                            operator=None)
```

Bases: *omegalpes.energy.units.energy\_units.EnergyUnit*

#### Description

Energy unit with a fixed power profile.

#### Attributes

- **p** : instantaneous power known by advance (kW)
- **energy\_type** : type of energy ('Electrical', 'Heat', ...)
- **operator** : stakeholder how owns the EnergyUnit

```
class omegalpes.energy.units.energy_units.SawtoothEnergyUnit (time, name,
                                                                flow_direction,
                                                                p_peak, p_low,
                                                                alpha_peak,
                                                                t_triangle,
                                                                t_sawtooth,
                                                                manda-
                                                                tory=True, start-
                                                                ing_cost=None,
                                                                operat-
                                                                ing_cost=None,
                                                                co2_out=None, en-
                                                                ergy_type=None,
                                                                operator=None)
```

Bases: *omegalpes.energy.units.energy\_units.ShiftableEnergyUnit*

```
class omegalpes.energy.units.energy_units.SeveralEnergyUnit (time, name,
                                                             fixed_power,
                                                             pmin=1e-05,
                                                             pmax=100000.0,
                                                             imaginary=False,
                                                             e_min=0,
                                                             e_max=1000000.0,
                                                             nb_unit_min=0,
                                                             nb_unit_max=None,
                                                             flow_direction='in',
                                                             starting_cost=None,
                                                             operat-
                                                             ing_cost=None,
                                                             max_ramp_up=None,
                                                             max_ramp_down=None,
                                                             co2_out=None, en-
                                                             ergy_type=None,
                                                             operator=None)
```

Bases: *omegalpes.energy.units.energy\_units.VariableEnergyUnit*

### Description

Energy unit based on a fixed power curve enabling to multiply several times (nb\_unit) the same power curve

Be careful, if imaginary == True, the solution may be imaginary as nb\_unit can be continuous. The accurate number of the power unit should be calculated later

### Attributs

- fixed\_power : fixed power curve

```
class omegalpes.energy.units.energy_units.ShiftableEnergyUnit (time, name: str,
                                                                flow_direction,
                                                                power_values,
                                                                mandatory=True,
                                                                co2_out=None,
                                                                start-
                                                                ing_cost=None,
                                                                operat-
                                                                ing_cost=None,
                                                                en-
                                                                ergy_type=None,
                                                                operator=None)
```

Bases: *omegalpes.energy.units.energy\_units.VariableEnergyUnit*

### Description

EnergyUnit with shiftable power profile.

### Attributs

- power\_values : power profile to shift (kW)
- **mandatory** [indicates if the power is mandatory][True] or not : False
- starting\_cost : cost of the starting of the EnergyUnit
- operating\_cost : cost of the operation (€/kW)
- energy\_type : type of energy ('Electrical', 'Heat', ...)

- operator : stakeholder how owns the EnergyUnit

```
class omegalpes.energy.units.energy_units.SquareEnergyUnit (time,          name,
                                                             p_square,  n_square,
                                                             t_between_sq,
                                                             t_square=1,
                                                             flow_direction='in',
                                                             starting_cost=None,
                                                             operating_cost=None,
                                                             co2_out=None,  en-
                                                             ergy_type=None,
                                                             operator=None)
```

Bases: *omegalpes.energy.units.energy\_units.VariableEnergyUnit*

```
class omegalpes.energy.units.energy_units.TriangleEnergyUnit (time,          name,
                                                                  flow_direction,
                                                                  p_peak,          al-
                                                                  pha_peak,
                                                                  t_triangle,  manda-
                                                                  tory=True,  start-
                                                                  ing_cost=None,
                                                                  operat-
                                                                  ing_cost=None,
                                                                  co2_out=None, en-
                                                                  ergy_type=None,
                                                                  operator=None)
```

Bases: *omegalpes.energy.units.energy\_units.ShiftableEnergyUnit*

```
class omegalpes.energy.units.energy_units.VariableEnergyUnit (time,          name,
                                                                  flow_direction='in',
                                                                  p_min=-10000.0,
                                                                  p_max=10000.0,
                                                                  e_min=-
                                                                  1000000.0,
                                                                  e_max=1000000.0,
                                                                  start-
                                                                  ing_cost=None,
                                                                  operat-
                                                                  ing_cost=None,
                                                                  min_time_on=None,
                                                                  min_time_off=None,
                                                                  max_ramp_up=None,
                                                                  max_ramp_down=None,
                                                                  co2_out=None,
                                                                  availabil-
                                                                  ity_hours=None,
                                                                  en-
                                                                  ergy_type=None,
                                                                  operator=None)
```

Bases: *omegalpes.energy.units.energy\_units.EnergyUnit*

## Consumption\_units module

**\*\* This module defines the consumption units\*\***

The `consumption_units` module defines various classes of consumption units, from generic to specific ones.

**It includes :**

- `ConsumptionUnit` : simple consumption unit. It inherits from `EnergyUnit`, its power flow direction is always 'in'.

**3 Objectives are also available :**

- minimize consumption, maximize consumption and minimize consumption costs.
- `FixedConsumptionUnit` : consumption with a fixed load profile. It inherits from `ConsumptionUnit`.
- `VariableConsumptionUnit` : consumption unit allowing for a variation of power between `pmin` et `pmax`. It inherits from `ConsumptionUnit`.

Copyright 2018 G2Elab / MAGE

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

```
class omegalpes.energy.units.consumption_units.ConsumptionUnit (time,      name,
                                                                p=None,
                                                                p_min=1e-05,
                                                                p_max=100000.0,
                                                                e_min=0,
                                                                e_max=1000000.0,
                                                                co2_out=None,
                                                                start-
                                                                ing_cost=None,
                                                                consump-
                                                                tion_cost=None,
                                                                min_time_on=None,
                                                                min_time_off=None,
                                                                max_ramp_up=None,
                                                                max_ramp_down=None,
                                                                availabil-
                                                                ity_hours=None,
                                                                en-
                                                                ergy_type=None,
                                                                opera-
                                                                tor=None)
```

Bases: `omegalpes.energy.units.energy_units.EnergyUnit`

**Description**

Simple Consumption unit

**Attributes**

- `p` : instantaneous power demand (kW)



- `pmax` : maximal instantaneous power demand (kW)
- `pmin` : minimal instantaneous power demand (kW)
- `energy_type` : type of energy ('Electrical', 'Heat', ...)
- `consumption_cost` : cost associated to the energy consumption
- `operator` : stakeholder how owns the consumption unit

**maximize\_consumption** (*weight=1*)

Parameters **weight** – Weight coefficient for the objective

**minimize\_consumption** (*weight=1*)

Parameters **weight** – Weight coefficient for the objective

**minimize\_consumption\_cost** (*weight=1*)

Parameters **weight** – Weight coefficient for the objective

```
class omegalpes.energy.units.consumption_units.FixedConsumptionUnit (time,  
                                                                    name,  
                                                                    p:    list  
                                                                    = None,  
                                                                    co2_out=None,  
                                                                    start-  
                                                                    ing_cost=None,  
                                                                    operat-  
                                                                    ing_cost=None,  
                                                                    en-  
                                                                    ergy_type=None,  
                                                                    opera-  
                                                                    tor=None)
```

Bases: `omegalpes.energy.units.energy_units.FixedEnergyUnit`, `omegalpes.energy.units.consumption_units.ConsumptionUnit`

### Description

Consumption unit with a fixed consumption profile.

### Attributes

- `p` : instantaneous power demand known in advance (kW)
- `energy_type` : type of energy ('Electrical', 'Heat', ...)
- `consumption_cost` : cost associated to the energy consumption
- `operator` : stakeholder how owns the consumption unit

```
class omegalpes.energy.units.consumption_units.SeveralConsumptionUnit (time,  
                                                                    name,  
                                                                    fixed_cons,  
                                                                    pmin=1e-  
                                                                    05,  
                                                                    pmax=100000.0,  
                                                                    e_min=0,  
                                                                    e_max=1000000.0,  
                                                                    nb_unit_min=0,  
                                                                    nb_unit_max=None,  
                                                                    co2_out=None,  
                                                                    start-  
                                                                    ing_cost=None,  
                                                                    op-  
                                                                    erat-  
                                                                    ing_cost=None,  
                                                                    max_ramp_up=None,  
                                                                    max_ramp_down=None,  
                                                                    en-  
                                                                    ergy_type=None,  
                                                                    op-  
                                                                    era-  
                                                                    tor=None)
```

Bases: `omegalpes.energy.units.consumption_units.VariableConsumptionUnit`,  
`omegalpes.energy.units.energy_units.SeveralEnergyUnit`

### Description

Consumption unit based on a fixed consumption curve enabling to multiply several times (nb\_unit) the same consumption curve

### Attributs

- `fixed_cons` : fixed consumption curve

```
class omegalpes.energy.units.consumption_units.SeveralImaginaryConsumptionUnit (time,  
                                         name,  
                                         fixed_cons,  
                                         pmin=1e-  
                                         05,  
                                         pmax=100000.0,  
                                         e_min=0,  
                                         e_max=1000000,  
                                         nb_unit_min=0,  
                                         nb_unit_max=None,  
                                         co2_out=None,  
                                         start-  
                                         ing_cost=None,  
                                         op-  
                                         er-  
                                         at-  
                                         ing_cost=None,  
                                         max_ramp_up=None,  
                                         max_ramp_down=None,  
                                         en-  
                                         ergy_type=None,  
                                         op-  
                                         er-  
                                         a-  
                                         tor=None)
```

Bases: `omegalpes.energy.units.consumption_units.VariableConsumptionUnit`,  
`omegalpes.energy.units.energy_units.SeveralEnergyUnit`

### Description

Consumption unit based on a fixed consumption curve enabling to multiply several times (nb\_unit) the same consumption curve. Be careful, the solution may be imaginary as nb\_unit can be continuous. The accurate number of the consumption unit should be calculated later

### Attributs

- fixed\_cons : fixed consumption curve

```
class omegalpes.energy.units.consumption_units.ShiftableConsumptionUnit (time,  
                                                                    name:  
                                                                    str,  
                                                                    power_values,  
                                                                    manda-  
                                                                    tory=True,  
                                                                    co2_out=None,  
                                                                    start-  
                                                                    ing_cost=None,  
                                                                    op-  
                                                                    er-  
                                                                    at-  
                                                                    ing_cost=None,  
                                                                    en-  
                                                                    ergy_type=None,  
                                                                    op-  
                                                                    er-  
                                                                    a-  
                                                                    tor=None)
```

Bases: `omegalpes.energy.units.energy_units.ShiftableEnergyUnit`, `omegalpes.energy.units.consumption_units.VariableConsumptionUnit`

### Description

Consumption unit with shiftable consumption profile.

### Attributes

- `power_values` : consumption profile to shift (kW)
- **mandatory** [indicates if the consumption is mandatory][True] or not : False
- `starting_cost` : cost of the starting of the consumption
- `operating_cost` : cost of the operation (€/kW)
- `energy_type` : type of energy ('Electrical', 'Heat', ...)
- `operator` : stakeholder how owns the consumption unit

```
class omegalpes.energy.units.consumption_units.SquareConsumptionUnit (time,  
                                                                    name,  
                                                                    p_square,  
                                                                    dura-  
                                                                    tion,  
                                                                    n_square,  
                                                                    t_between_sq,  
                                                                    co2_out=None,  
                                                                    start-  
                                                                    ing_cost=None,  
                                                                    operat-  
                                                                    ing_cost=None,  
                                                                    en-  
                                                                    ergy_type=None,  
                                                                    opera-  
                                                                    tor=None)
```

Bases: `omegalpes.energy.units.energy_units.SquareEnergyUnit`, `omegalpes.energy.units.consumption_units.VariableConsumptionUnit`

### Description

**Consumption unit with a fixed value and fixed duration.** >> Only the time of beginning can be modified >> Operation can be mandatory or not

#### Attributes

- `p` : instantaneous power consumption (kW)
- `duration` : duration of the power delivery (hours)
- `mandatory` : indicates if the power delivery is mandatory or not
- `energy_type` : type of energy ('Electrical', 'Heat', ...)
- `consumption_cost` : cost associated to the energy consumption
- `operator` : stakeholder how owns the consumption unit

```
class omegalpes.energy.units.consumption_units.VariableConsumptionUnit (time,
                                                                    name,
                                                                    pmin=1e-
                                                                    05,
                                                                    pmax=100000.0,
                                                                    e_min=0,
                                                                    e_max=1000000.0,
                                                                    co2_out=None,
                                                                    start-
                                                                    ing_cost=None,
                                                                    op-
                                                                    er-
                                                                    at-
                                                                    ing_cost=None,
                                                                    min_time_on=None,
                                                                    min_time_off=None,
                                                                    max_ramp_up=None,
                                                                    max_ramp_down=None,
                                                                    en-
                                                                    ergy_type=None,
                                                                    op-
                                                                    era-
                                                                    tor=None)

Bases:    omegalpes.energy.units.energy_units.VariableEnergyUnit, omegalpes.
energy.units.consumption_units.ConsumptionUnit
```

#### Description

Consumption unit with a variation of power between `pmin` et `pmax`.

#### Attributes

- `pmax` : maximal instantaneous power consumption (kW)
- `pmin` : minimal instantaneous power consumption (kW)
- `energy_type` : type of energy ('Electrical', 'Heat', ...)
- `operator` : stakeholder how owns the consumption unit

## Production\_units module

**\*\* This module defines the production units\*\***

The `production_units` module defines various kinds of production units with associated attributes and methods.

**It includes :**

- **ProductionUnit** : simple production unit inheriting from **EnergyUnit** and with an outer flow direction. The outside co2 emissions, the starting cost, the operating cost, the minimal operating time, the minimal non-operating time, the maximal increasing ramp and the maximal decreasing ramp can be filled.

**Objectives are also available :**

- minimize starting cost, operating cost, total cost
- minimize production, co2\_emissions, time of use
- maximize production
- **FixedProductionUnit** : Production unit with a fixed production profile.
- **VariableProductionUnit** : Production unit with a variation of power between pmin et pmax.

**And also :**

- **SeveralProductionUnit**: Production unit based on a fixed production curve enabling to multiply several times (nb\_unit) the same production curve
- **SeveralImaginaryProductionUnit**: Production unit based on a fixed production curve enabling to multiply several times (nb\_unit) the same production curve. Be careful, the solution may be imaginary as nb\_unit can be continuous. The accurate number of the production unit should be calculated later
- **SquareProductionUnit**: Production unit with a fixed value and fixed duration.
- **ShiftableProductionUnit**: Production unit with shiftable production profile.

Copyright 2018 G2Elab / MAGE

Licensed under the Apache License, Version 2.0 (the “License”); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

```
class omegalpes.energy.units.production_units.FixedProductionUnit (time, name:
                                                                    str,      p:
                                                                    list = None,
                                                                    co2_out=None,
                                                                    start-
                                                                    ing_cost=None,
                                                                    operat-
                                                                    ing_cost=None,
                                                                    en-
                                                                    ergy_type=None,
                                                                    opera-
                                                                    tor=None)

Bases: omegalpes.energy.units.energy_units.FixedEnergyUnit, omegalpes.energy.
units.production_units.ProductionUnit
```

**Description**

Production unit with a fixed production profile.

**Attributes**

- p : instantaneous power production known by advance (kW)

- `energy_type` : type of energy ('Electrical', 'Heat', ...)
- `operator` : stakeholder how owns the production unit

```
class omegalpes.energy.units.production_units.ProductionUnit (time,           name,
                                                             p=None,
                                                             p_min=1e-05,
                                                             p_max=100000.0,
                                                             e_min=0,
                                                             e_max=1000000.0,
                                                             co2_out=None,
                                                             start-
                                                             ing_cost=None,
                                                             operat-
                                                             ing_cost=None,
                                                             min_time_on=None,
                                                             min_time_off=None,
                                                             max_ramp_up=None,
                                                             max_ramp_down=None,
                                                             availabil-
                                                             ity_hours=None,
                                                             en-
                                                             ergy_type=None,
                                                             operator=None)
```

Bases: `omegalpes.energy.units.energy_units.EnergyUnit`

### Description

Simple Production unit

### Attributes

- `co2_out`: outside co2 emissions
- `starting_cost`: the starting cost
- `operating_cost`: the operating cost
- `min_time_on` : the minimal operating time
- `min_time_off` : the minimal non-operating time
- `max_ramp_up` : the maximal increasing ramp
- `max_ramp_down` ; the maximal decreasing ramp

**maximize\_production** (*weight=1*)

Parameters **weight** – Weight coefficient for the objective

**minimize\_production** (*weight=1*)

Parameters **weight** – Weight coefficient for the objective

```
class omegalpes.energy.units.production_units.SeveralImaginaryProductionUnit (time,  
name,  
fixed_prod,  
pmin=1e-  
05,  
pmax=100000.0,  
e_min=0,  
e_max=1000000.0,  
nb_unit_min=0,  
nb_unit_max=None,  
co2_out=None,  
start-  
ing_cost=None,  
op-  
er-  
at-  
ing_cost=None,  
max_ramp_up=None,  
max_ramp_down=None,  
en-  
ergy_type=None,  
op-  
er-  
a-  
tor=None)
```

Bases: `omegalpes.energy.units.production_units.VariableProductionUnit`,  
`omegalpes.energy.units.energy_units.SeveralEnergyUnit`

### Description

Production unit based on a fixed production curve enabling to multiply several times (`nb_unit`) the same production curve. Be careful, the solution may be imaginary as `nb_unit` can be continuous. The accurate number of the production unit should be calculated later

### Attributes

- `fixed_prod` : fixed production curve



```
class omegalpes.energy.units.production_units.SeveralProductionUnit (time,  
                                                                    name,  
                                                                    fixed_prod,  
                                                                    pmin=1e-  
                                                                    05,  
                                                                    pmax=100000.0,  
                                                                    e_min=0,  
                                                                    e_max=1000000.0,  
                                                                    nb_unit_min=0,  
                                                                    nb_unit_max=None,  
                                                                    co2_out=None,  
                                                                    start-  
                                                                    ing_cost=None,  
                                                                    operat-  
                                                                    ing_cost=None,  
                                                                    max_ramp_up=None,  
                                                                    max_ramp_down=None,  
                                                                    en-  
                                                                    ergy_type=None,  
                                                                    opera-  
                                                                    tor=None)
```

Bases: *omegalpes.energy.units.production\_units.VariableProductionUnit*,  
*omegalpes.energy.units.energy\_units.SeveralEnergyUnit*

#### Description

Production unit based on a fixed production curve enabling to multiply several times (nb\_unit) the same production curve nb\_unit is an integer variable

#### Attributes

- fixed\_prod : fixed production curve

```
class omegalpes.energy.units.production_units.ShiftableProductionUnit (time,  
                                                                    name:  
                                                                    str,  
                                                                    power_values,  
                                                                    manda-  
                                                                    tory=True,  
                                                                    co2_out=None,  
                                                                    start-  
                                                                    ing_cost=None,  
                                                                    op-  
                                                                    erat-  
                                                                    ing_cost=None,  
                                                                    en-  
                                                                    ergy_type=None,  
                                                                    op-  
                                                                    era-  
                                                                    tor=None)
```

Bases: *omegalpes.energy.units.energy\_units.ShiftableEnergyUnit*, *omegalpes.energy.units.production\_units.VariableProductionUnit*

#### Description

Production unit with shiftable production profile.

#### Attributes

- `power_values` : production profile to shift (kW)
- `mandatory` : indicates if the production is mandatory : True or not : False
- `starting_cost` : cost of the starting of the production
- `operating_cost` : cost of the operation (€/kW)
- `energy_type` : type of energy ('Electrical', 'Heat', ...)
- `operator` : stakeholder how owns the production unit

```
class omegalpes.energy.units.production_units.SquareProductionUnit (time,  
                                                                    name,  
                                                                    p_square,  
                                                                    duration,  
                                                                    n_square,  
                                                                    t_between_sq,  
                                                                    co2_out=None,  
                                                                    start-  
                                                                    ing_cost=None,  
                                                                    operat-  
                                                                    ing_cost=None,  
                                                                    en-  
                                                                    ergy_type=None,  
                                                                    opera-  
                                                                    tor=None)
```

Bases: `omegalpes.energy.units.energy_units.SquareEnergyUnit`, `omegalpes.energy.units.production_units.VariableProductionUnit`

### Description

Production unit with a fixed value and fixed duration. Only the time of beginning can be modified  
Operation can be mandatory or not

### Attributes

- `p` : instantaneous power production (kW)
- `duration` : duration of the power delivery (hours)
- `mandatory` : indicates if the power delivery is mandatory or not
- `energy_type` : type of energy ('Electrical', 'Heat', ...)
- `operator` : stakeholder how owns the production unit

```

class omegalpes.energy.units.production_units.VariableProductionUnit (time,
                                                                    name,
                                                                    pmin=1e-
                                                                    05,
                                                                    pmax=100000.0,
                                                                    e_min=0,
                                                                    e_max=1000000.0,
                                                                    co2_out=None,
                                                                    start-
                                                                    ing_cost=None,
                                                                    operat-
                                                                    ing_cost=None,
                                                                    min_time_on=None,
                                                                    min_time_off=None,
                                                                    max_ramp_up=None,
                                                                    max_ramp_down=None,
                                                                    en-
                                                                    ergy_type=None,
                                                                    opera-
                                                                    tor=None)
omegalpes.

Bases:    omegalpes.energy.units.energy_units.VariableEnergyUnit,
          energy.units.production_units.ProductionUnit

```

### Description

Production unit with a variation of power between pmin et pmax.

### Attributes

- pmax : maximal instantaneous power production (kW)
- pmin : minimal instantaneous power production (kW)
- energy\_type : type of energy ('Electrical', 'Heat', ...)
- operator : stakeholder how owns the production unit

## Conversion\_units module

**\*\* This module defines the conversion units, with at least a production unit and a consumption unit and using one or several energy types\*\***

The conversion\_units module defines various classes of conversion units, from generic to specific ones.

### It includes :

- ConversionUnit : simple conversion unit. It inherits from Unit.
- ElectricalToHeatConversionUnit : Electrical to heat Conversion unit with an electricity consumption and a heat production linked by and electrical to heat ratio. It inherits from ConversionUnit
- HeatPump : Simple Heat Pump with an electricity consumption, a heat production and a heat consumption. It has a theoretical coefficient of performance COP and inherits from ConversionUnit.

Copyright 2018 G2Elab / MAGE

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

```
class omegalpes.energy.units.conversion_units.ConversionUnit (time,          name,
                                                             prod_units=None,
                                                             cons_units=None,
                                                             operator=None)
```

Bases: *omegalpes.general.optimisation.units.Unit*

### Description

Simple Conversion unit

### Attributes

- time : TimeUnit describing the studied time period
- prod\_units : list of the production units
- cons\_units : list of the consumption units
- operator : stakeholder who owns the conversion unit
- poles : dictionary of the poles of the conversion unit

```
class omegalpes.energy.units.conversion_units.ElectricalToHeatConversionUnit (time,
                                                                              name,
                                                                              pmin_in_elec=1e-05,
                                                                              pmax_in_elec=100,
                                                                              p_in_elec=None,
                                                                              pmin_out_heat=1e-05,
                                                                              pmax_out_heat=100,
                                                                              p_out_heat=None,
                                                                              elec_to_heat_ratio=operator.a-
                                                                              tor=None)
```

Bases: *omegalpes.energy.units.conversion\_units.ConversionUnit*

### Description

Electrical to heat Conversion unit with an electricity consumption and a heat production

### Attributes

- heat\_production\_unit : heat production unit (heat output)
- elec\_consumption\_unit : electricity consumption unit (electrical input)
- conversion : Dynamic Constraint linking the electrical input to the heat output through the electrical to heat ratio

```

class omegalpes.energy.units.conversion_units.HeatPump (time, name,
                                                         pmin_in_elec=1e-05,
                                                         pmax_in_elec=100000.0,
                                                         p_in_elec=None,
                                                         pmin_in_heat=1e-05,
                                                         pmax_in_heat=100000.0,
                                                         p_in_heat=None,
                                                         pmin_out_heat=1e-05,
                                                         pmax_out_heat=100000.0,
                                                         p_out_heat=None, cop=3,
                                                         losses=0, operator=None)

```

Bases: `omegalpes.energy.units.conversion_units.ConversionUnit`

### Description

Simple Heat Pump with an electricity consumption, a heat production and a heat consumption. It has a theoretical coefficient of performance COP and inherits from ConversionUnit.

### Attributes

- `heat_production_unit` : heat production unit (condenser)
- `elec_consumption_unit` : electricity consumption unit (electrical input)
- `heat_consumption_unit` : heat consumption unit (evaporator)
- `COP` : Quantity describing the coefficient of performance of the heat pump
- `conversion` : Dynamic Constraint linking the electrical input to the heat output through the electrical to heat ratio
- `power_flow` : Dynamic constraint linking the heat output to the electrical and heat inputs in relation to the losses.

## Storage\_units module

**\*\* This module defines the storage units\*\***

The storage\_units module defines various kinds of storage units with associated attributes and methods, from simple to specific ones.

### It includes :

- `StorageUnit` : simple storage unit inheriting from EnergyUnit, with storage specific attributes. It includes the objective “minimize capacity”.
- `Thermocline storage` : a thermal storage that need to cycle (i.e. reach SOC\_max) every period of Tcycle

Copyright 2018 G2Elab / MAGE

Licensed under the Apache License, Version 2.0 (the “License”); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

```
class omegalpes.energy.units.storage_units.StorageUnit (time, name, pc_min=1e-05, pc_max=100000.0, pd_min=1e-05, pd_max=100000.0, capacity=None, e_0=None, e_f=None, soc_min=0, soc_max=1, eff_c=1, eff_d=1, self_disch=0, self_disch_t=0, ef_is_e0=False, cycles=None, energy_type=None, operator=None)
```

Bases: *omegalpes.energy.units.energy\_units.VariableEnergyUnit*

### Description

Simple Storage unit

### Attributes

- capacity (Quantity): maximal energy that can be stored
- e (Quantity): energy at time t in the storage
- set\_soc\_min (DynamicConstraint): constraining the energy to be above the value : soc\_min\*capacity
- set\_soc\_max (DynamicConstraint): constraining the energy to be below the value : soc\_max\*capacity
- pc (Quantity) : charging power
- pd (Quantity) : discharging power
- u\_c (Quantity) : binary variable describing the charge of the storage unit : 0 : Not charging & 1 : charging
- calc\_e (DynamicConstraint) : energy calculation at time t ; relation power/energy
- calc\_p (DynamicConstraint) : power calculation at time t ; power flow equals charging power minus discharging power
- on\_off\_stor (DynamicConstraint) : making u[t] matching with storage modes (on/off)
- def\_max\_charging (DynamicConstraint) : defining the max charging power, avoiding charging and discharging at the same time
- def\_max\_discharging (DynamicConstraint) : defining the max discharging power, avoiding charging and discharging at the same time
- def\_min\_charging (DynamicConstraint) : defining the min charging power, avoiding charging and discharging at the same time
- def\_min\_discharging (DynamicConstraint) : defining the min discharging power, avoiding charging and discharging at the same time
- set\_e\_0 (ExternalConstraint) : set the energy state for t=0
- e\_f (Quantity) : energy in the storage at the end of the time horizon, i.e. after the last time step
- e\_f\_min (Constraint) : e\_f value is constrained above soc\_min\*capacity
- e\_f\_max (Constraint) : e\_f value is constrained below soc\_max\*capacity
- set\_e\_f (Constraint) : when e\_f is given, it is set in the same way the energy is, but after the last time step
- calc\_e\_f (Constraint) : when e\_f is not given, it is calculated in the same way the energy is, but after the last time step

- `ef_is_e0` (ExternalConstraint) : Imposing  $ef=e0$  on the time period.
- `cycles` (ExternalDynamicConstraint) : setting a cycle constraint  $e[t] = e[t+cycles/dt]$

**minimize\_capacity** (*weight=1*)

**Parameters** `weight` – Weight coefficient for the objective

```
class omegalpes.energy.units.storage_units.ThermoclineStorage(time,          name,
                                                             pc_min=1e-05,
                                                             pc_max=100000.0,
                                                             pd_min=1e-05,
                                                             pd_max=100000.0,
                                                             capacity=None,
                                                             e_0=None,
                                                             e_f=None,
                                                             soc_min=0,
                                                             soc_max=1,
                                                             eff_c=1, eff_d=1,
                                                             self_disch=0,
                                                             Tcycl=120,
                                                             ef_is_e0=False,
                                                             operator=None)
```

Bases: `omegalpes.energy.units.storage_units.StorageUnit`

### Description

Class ThermoclineStorage : class defining a thermocline heat storage, inheriting from StorageUnit.

### Attributes

- `is_soc_max` (Quantity) : indicating if the storage is fully charged 0:No 1:Yes
- `def_is_soc_max_inf` (DynamicConstraint) : setting the right value for `is_soc_max`
- `def_is_soc_max_sup` (DynamicConstraint) : setting the right value for `is_soc_max`
- `force_soc_max` (ExtDynConstraint) : The energy has to be at least once at its maximal value during the period `Tcycl`.

## Energy\_nodes module

**\*\* This module defines the energy nodes that will allow energy transmission between the various energy units and conversion units \*\***

The `energy_node` module includes the `EnergyNode` class for energy transmission between production, consumption, conversion and storage. Defining several energy nodes and exporting/importing energy between them can also allow for a better demarcation of the energy system.

Copyright 2018 G2Elab / MAGE

Licensed under the Apache License, Version 2.0 (the “License”); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

```
class omegalpes.energy.energy_nodes.EnergyNode (time, name, energy_type=None, operator=None)
```

Bases: *omegalpes.general.optimisation.units.Unit*

This class defines an energy node.

```
add_connected_energy_unit (unit)
```

Add an EnergyUnit to the connected\_units list

```
add_pole (pole: omegalpes.energy.io.poles.Epole) → None
```

Add an energy pole to the poles\_list

**Parameters** *pole* – Epole

```
connect_units (*units)
```

Connecting all EnergyUnit to the EnergyNode

**Parameters** *units* (*list*) – EnergyUnits connected to the EnergyNode

```
create_export (node, export_min, export_max)
```

Create the export from the EnergyNode (self) to the EnergyNode (node)

**Parameters**

- **node** – EnergyNode to whom power can be exported
- **export\_min** – Minimal value of exported power when there is export
- **export\_max** – Maximal value of exported power when there is export

**Returns** Quantity that defines the power exported

```
export_to_node (node, export_min=1e-05, export_max=100000.0)
```

Add an export of power from the node to another node

**Parameters**

- **node** – EnergyNode to whom power can be exported
- **export\_min** – Minimal value of exported power when there is export
- **export\_max** – Maximal value of exported power when there is export

```
get_connected_energy_units
```

Return the list of connected EnergyUnits in the EnergyNode

```
get_exports
```

Return the list of exports to the EnergyNode

```
get_flows
```

Get all the power flows of the energy node :rtype: list :return: list of power flows

```
get_imports
```

Return the list of imports to the EnergyNode

```
get_poles
```

Return the list of energy poles in the EnergyNode

```
import_from_node (node, import_min=1e-05, import_max=100000.0)
```

**Parameters**

- **node** – EnergyNode from whom power can be imported
- **import\_min** – Minimal value of imported power when there is import
- **import\_max** – Maximal value of imported power when there is import



```

is_export_flow (flow)
    Get if the power flow is an export or not

is_import_flow (flow)
    Get if the power flow is an import or not

set_power_balance ()
    Set the power balance equation for the EnergyNode

```

## Poles module

**\*\* This module defines inputs and outputs of as poles\*\***

**The poles module includes :**

- FlowPole : this class defines a pole with a directed flow (in or out)
- EPole : this class define an energy pole

Copyright 2018 G2Elab / MAGE

Licensed under the Apache License, Version 2.0 (the “License”); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

```

class omegalpes.energy.io.poles.Epole (p, direction, energy_type=None)
    Bases: omegalpes.energy.io.poles.FlowPole

    ** Description **

    Definition of an energetic pole, power and power flow direction convention ‘in’ or ‘out’

class omegalpes.energy.io.poles.FlowPole (flow='flow', direction='in')
    Bases: dict

    Interface for basics flux poles

```

The **actor classes** enables to build the energy model considering pre-defined stakeholders’ constraints and objectives

## 2.4.2 actor package

The actor modelling is based on a main actor class defined on the actor module. Then, the actors are divided in two categories: the “operator actors” who operates energy units and the “regulator actors” who unable to create regulation constraints.

- *Actor module*
- *Operator\_actors module*
- *Consumer\_actors module*
- *Producer\_actors module*
- *Consumer\_producer\_actors module*

- *Regulator\_actors module*

## Actor module

**\*\* This modules define the basic Actor object \*\***

### Few methods are available:

- `add_external_constraint`
- `add_external_dynamic_constraint`
- `add_objective`

Copyright 2018 G2Elab / MAGE

Licensed under the Apache License, Version 2.0 (the “License”); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

**class** `omegalpes.actor.actor.Actor` (*name*)

Bases: *omegalpes.general.optimisation.units.Unit*

### Description

Actor class is the basic class to model an actor. The basic actor is defined by its name and description. An actor is then defined by its constraints and objectives.

### Attributes

- `description` : description as an Actor Unit

**add\_external\_constraint** (*cst\_name, exp*)

Enable to add an external constraint linked with an actor

#### Parameters

- **cst\_name** – name of the constraint
- **exp** – expression of the constraint

**add\_external\_dynamic\_constraint** (*cst\_name, exp\_t, t\_range='for t in time.I'*)

Enable to add an external dynamic constraint linked with an actor. A dynamic constraint changes over time

#### Parameters

- **cst\_name** – name of the constraint
- **exp** – expression of the constraint depending on the time
- **t\_range** – expression of time for the constraint

**add\_objective** (*obj\_name, exp*)

Enable to add an objective linked with an actor

#### Parameters

- **obj\_name** – name of the objective

- **exp** – expression of the objective

## Operator\_actors module

**\*\* This module defines the operator\_actor and its scope of responsibility \*\***

Copyright 2018 G2Elab / MAGE

Licensed under the Apache License, Version 2.0 (the “License”); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

```
class omegAlpes.actor.operator_actors.operator_actors.OperatorActor (name,  
                                                                    oper-  
                                                                    ated_unit_type_tuple,  
                                                                    oper-  
                                                                    ated_unit_list=None,  
                                                                    oper-  
                                                                    ated_node_list=None)
```

Bases: *omegAlpes.actor.actor.Actor*

### Description

OperatorActor class inherits from the the basic class Actor. It enables one to model an actor who operates energy units which is part of its scope of responsibility. An operator actor has objectives and constraints which are linked to the energy units he operates.

### Attributes

- **name** : name of the actor
- **operated\_unit\_list**: list of the energy units operated by the actor or more precisely in its scope of responsibility

## Consumer\_actors module

**\*\* This module describes the Consumer actor \*\***

Few objectives and constraints are available. Objectives :

- **maximize\_consumption**
- **minimize\_consumption**
- **minimize\_co2\_consumption**
- **minimize\_consumption\_costs**

### Constraints :

- **energy\_consumption\_minimum**
- **energy\_consumption\_maximum**
- **power\_consumption\_minimum**
- **power\_consumption\_maximum**

Copyright 2018 G2Elab / MAGE

Licensed under the Apache License, Version 2.0 (the “License”); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

```
class omegalpes.actor.operator_actors.consumer_actors.Consumer(name,      oper-
                                                                ated_unit_list,
                                                                oper-
                                                                ated_node_list=None)

Bases: omegalpes.actor.operator_actors.operator_actors.OperatorActor
```

### Description

Consumer class inherits from the the class OperatorActor. It enables one to model a consumer actor.

**energy\_consumption\_maximum** (*max\_e\_tot*, *cst\_operated\_unit\_list*=None)

To create the actor constraint of a maximum of energy consumption.

#### Parameters

- **max\_e\_tot** – Maximum of the total energy consumption over the study period
- **cst\_operated\_unit\_list** – List of units on which the constraint will be applied.  
Might be empty.

**energy\_consumption\_minimum** (*min\_e\_tot*, *cst\_operated\_unit\_list*=None)

To create the actor constraint of a minimum of energy consumption.

#### Parameters

- **min\_e\_tot** – Minimum of the total energy consumption over the study period
- **cst\_operated\_unit\_list** – List of units on which the constraint will be applied.  
Might be empty.

**maximize\_consumption** (*obj\_operated\_unit\_list*=None, *weight*=1)

To create the objective in order to maximize the consumption of the consumer’s units (all or part of them).

#### Parameters

- **obj\_operated\_unit\_list** – List of units on which the objective will be applied.  
Might be empty
- **weight** – Weight coefficient for the objective

**minimize\_co2\_consumption** (*obj\_operated\_unit\_list*=None, *weight*=1)

To create the objective in order to minimize the co2 emissions due to the consumer’s units (all or part of them).

#### Parameters

- **obj\_operated\_unit\_list** – List of units on which the objective will be applied.  
Might be empty
- **weight** – Weight coefficient for the objective

**minimize\_consumption** (*obj\_operated\_unit\_list*=None, *weight*=1)

To create the objective in order to minimize the consumption of the consumer’s units (all or part of them).

**Parameters**

- **obj\_operated\_unit\_list** – List of units on which the objective will be applied.  
Might be empty
- **weight** – Weight coefficient for the objective

**minimize\_consumption\_cost** (*obj\_operated\_unit\_list=None, weight=1*)

To create the objective in order to minimize the expenses due to the consumer's units (all or part of them).

**Parameters**

- **obj\_operated\_unit\_list** – List of units on which the objective will be applied.  
Might be empty
- **weight** – Weight coefficient for the objective

**power\_consumption\_maximum** (*max\_p, time, cst\_operated\_unit\_list=None*)

To create the actor constraint of a maximum of power consumption.

**Parameters**

- **max\_p** – Maximum of the power consumption. May be an int, float or a list with the size of the period study
- **time** – period of the study
- **cst\_operated\_unit\_list** – List of units on which the constraint will be applied.  
Might be empty.

**power\_consumption\_minimum** (*min\_p, time, cst\_operated\_unit\_list=None*)

To create the actor constraint of a minimum of power consumption.

**Parameters**

- **min\_p** – Minimum of the power consumption. May be an int, float or a list with the size of the period study
- **time** – period of the study
- **cst\_operated\_unit\_list** – List of units on which the constraint will be applied.  
Might be empty.

**Producer\_actors module**

**\*\* This module describes the Producer actor \*\***

Few objectives and constraints are available. Objectives :

- maximize\_production
- minimize\_production
- minimize\_time\_of\_use
- minimize\_co2\_emissions
- minimize\_costs
- minimize\_operating\_cost
- minimize\_starting\_cost

**Constraints :**

- energy\_production\_minimum

- `energy_production_maximum`
- `power_production_minimum`
- `power_production_maximum`

Copyright 2018 G2Elab / MAGE

Licensed under the Apache License, Version 2.0 (the “License”); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

```
class omegalpes.actor.operator_actors.producer_actors.Producer(name, operated_unit_list,
                                                                operated_node_list=None)
Bases: omegalpes.actor.operator_actors.operator_actors.OperatorActor
```

**Description** Producer class inherits from the the class `OperatorActor`. It enables one to model an energy producer actor.

**energy\_production\_maximum** (*max\_e\_tot*, *cst\_operated\_unit\_list=None*)

To create the actor constraint of a maximum of energy production.

#### Parameters

- **max\_e\_tot** – Maximum of the total energy production over the period of the study
- **cst\_operated\_unit\_list** – List of units on which the constraint will be applied. Might be empty.

**energy\_production\_minimum** (*min\_e\_tot*, *cst\_operated\_unit\_list=None*)

To create the actor constraint of a minimum of energy production.

#### Parameters

- **min\_e\_tot** – Minimum of the total energy production over the period of the study
- **cst\_operated\_unit\_list** – List of units on which the constraint will be applied. Might be empty.

**maximize\_production** (*obj\_operated\_unit\_list=None*, *weight=1*)

To create the objective in order to maximize the production of the producer’s units (all or part of them).

#### Parameters

- **obj\_operated\_unit\_list** – List of units on which the objective will be applied. Might be empty.
- **weight** – Weight coefficient for the objective

**minimize\_co2\_emissions** (*obj\_operated\_unit\_list=None*, *weight=1*)

To create the objective in order to minimize the co2 emissions of the producer’s units (all or part of them). based on the quantity “co2\_emission”

#### Parameters

- **obj\_operated\_unit\_list** – list of the operated energy units on which the objective will be applied

- **weight** – weight of the objective

**minimize\_costs** (*obj\_operated\_unit\_list=None, weight=1*)

To create the objective in order to minimize the cost of the producer's units (all or part of them).

#### Parameters

- **obj\_operated\_unit\_list** – list of the operated energy units on which the objective will be applied
- **weight** – weight of the objective

**minimize\_operating\_cost** (*obj\_operated\_unit\_list=None, weight=1*)

To create the objective in order to minimize the operating costs of the producer's units (all or part of them).

#### Parameters

- **obj\_operated\_unit\_list** – list of the operated energy units on which the objective will be applied
- **weight** – weight of the objective

**minimize\_production** (*obj\_operated\_unit\_list=None, weight=1*)

To create the objective in order to minimize the production of the producer's units (all or part of them).

#### Parameters

- **obj\_operated\_unit\_list** – List of units on which the objective will be applied. Might be empty.
- **weight** – Weight coefficient for the objective

**minimize\_starting\_cost** (*obj\_operated\_unit\_list=None, weight=1*)

To create the objective in order to minimize the starting costs of the producer's units (all or part of them).

#### Parameters

- **obj\_operated\_unit\_list** – list of the operated energy units on which the objective will be applied
- **weight** – weight of the objective

**minimize\_time\_of\_use** (*obj\_operated\_unit\_list=None, weight=1*)

To create the objective in order to minimize the time of use of the producer's units (all or part of them).

#### Parameters

- **obj\_operated\_unit\_list** – List of units on which the objective will be applied. Might be empty.
- **weight** – Weight coefficient for the objective

**power\_production\_maximum** (*max\_p, time, cst\_operated\_unit\_list=None*)

To create the actor constraint of a maximum of power production.

#### Parameters

- **max\_p** – Minimum of the power production. May be an int, float or a list with the size of the period study
- **time** – period of the study
- **cst\_operated\_unit\_list** – List of units on which the constraint will be applied. Might be empty.

**power\_production\_minimum** (*min\_p, time, cst\_operated\_unit\_list=None*)

To create the constraint of a minimum of power production.

### Parameters

- **min\_p** – Minimum of the power production. May be an int, float or a list with the size of the period study
- **time** – period of the study
- **cst\_operated\_unit\_list** – List of units on which the constraint will be applied. Might be empty.

## Consumer\_producer\_actors module

**\*\* This module describes the Prosumer (producer and consumer) actor \*\***

Copyright 2018 G2Elab / MAGE

Licensed under the Apache License, Version 2.0 (the “License”); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

```
class omegalpes.actor.operator_actors.consumer_producer_actors.Prosumer (name,  
oper-  
er-  
ated_consumption_unit_list,  
op-  
er-  
ated_production_unit_list,  
op-  
er-  
ated_node_list=None)  
  
Bases: omegalpes.actor.operator_actors.consumer_actors.Consumer, omegalpes.  
actor.operator_actors.producer_actors.Producer
```

**Description** Prosumer class inherits from the the class OperatorActor, Consumer and Producer. It enables one to model an actor which is at the same time an energy producer and consumer

```
maximize_conso_prod_match (time, obj_operated_consumption_unit_list=None,  
obj_operated_production_unit_list=None, weight=1)
```

To create the objective in order to match at each time the consumption with the local production of the prosumer’s units (all or part of them).

### Parameters

- **obj\_operated\_consumption\_unit\_list** – List of consumption units on which the objective will be applied. Might be empty.
- **obj\_operated\_production\_unit\_list** – List of production units on which the objective will be applied. Might be empty.
- **weight** – Weight coefficient for the objective



```
maximize_selfconsumption_rate (time, obj_operated_production_unit_list=None,
                                obj_operated_consumption_unit_list=None,
                                obj_operated_selfconsummed_production_export_list=None,
                                obj_operated_selfconsummed_production_unit_list=None,
                                weight=1)
```

To create the objective in order to maximize the selfconsumption rate of the prosumer's units (all or part of them) WHILE maximizing the load matching selfconsummed production is calculated with the export nodes.

Selfconsumption rate = selfconsummed production / total production

#### Parameters

- **obj\_operated\_production\_unit\_list** – List of production units on which the objective will be applied. Might be empty.
- **obj\_operated\_selfconsummed\_production\_export\_list** – List of production exports from nodes on which the objective will be applied. Might be empty.
- **obj\_operated\_selfconsummed\_production\_unit\_list** – List of production units on which the objective will be applied. Might be empty.
- **weight** – Weight coefficient for the objective

```
maximize_selfproduction_rate (time, obj_operated_consumption_unit_list=None,
                                obj_operated_production_unit_list=None,
                                obj_operated_selfproduced_consumption_export_list=None,
                                obj_operated_selfproduced_consumption_unit_list=None,
                                weight=1)
```

To create the objective in order to maximize the selfproduction rate of the prosumer's units (all or part of them) WHILE maximizing the load matching selfproduced consumption may required export nodes

Selfproduction rate = selfproduced consumption / total consumption

#### Parameters

- **obj\_operated\_consumption\_unit\_list** – List of consumption units on which the objective will be applied. Might be empty.
- **obj\_operated\_selfproduced\_consumption\_export\_list** – List of production exports from nodes on which the objective will be applied. Might be empty.
- **obj\_operated\_selfproduced\_consumption\_unit\_list** – List of production units on which the objective will be applied. Might be empty.
- **weight** – Weight coefficient for the objective

```
class omegalpes.actor.operator_actors.consumer_producer_actors.Supplier (name,
                                                                    oper-
                                                                    er-
                                                                    ated_consumption_unit_list
                                                                    oper-
                                                                    er-
                                                                    ated_production_unit_list)
```

Bases: `omegalpes.actor.operator_actors.consumer_actors.Consumer`, `omegalpes.actor.operator_actors.producer_actors.Producer`

**Description** Supplier class inherits from the the class OperatorActor, Consumer and Producer. It enables one to model a supplier.

## Regulator\_actors module

**\*\* This module defines the operator\_actor and its scope of responsibility \*\***

**One constraint is available :**

- `co2_emission_maximum`

Copyright 2018 G2Elab / MAGE

Licensed under the Apache License, Version 2.0 (the “License”); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

**class** `omegalpes.actor.regulator_actors.regulator_actors.LocalAuthorities(name)`  
Bases: `omegalpes.actor.regulator_actors.regulator_actors.RegulatorActor`

**Description** LocalAuthorities class inherits from the basic class RegulatorActor. It focuses on local Authorities constraints

**class** `omegalpes.actor.regulator_actors.regulator_actors.PublicAuthorities(name)`  
Bases: `omegalpes.actor.regulator_actors.regulator_actors.RegulatorActor`

**Description** PublicAuthorities class inherits from the basic class RegulatorActor. It focuses on local Authorities constraints

**class** `omegalpes.actor.regulator_actors.regulator_actors.RegulatorActor(name)`  
Bases: `omegalpes.actor.actor.Actor`

### Description

RegulatorActor class inherits from the the basic class Actor. It enables one to model an actor who can add constraints on all energy units of the study case.

### Attributes

- `name` : name of the actor

**`co2_emission_maximum(max_co2, time, cst_production_list)`**  
To create the actor constraint of a maximum of CO2 emission.

### Parameters

- **`max_co2`** – Minimum of the CO2 emission. May be an int, float or a list with the size of the period study
- **`time`** – period of the study
- **`cst_unit_list`** – List of units on which the constraint will be applied.

The **general classes** helps to build the units, the model and to plot the results

## 2.4.3 general package

Please, have a look to the following general modules

- *Elements module*
- *Model module*
- *Units module*
- *Plots module*
- *Time module*
- *Utils module*

## Elements module

This module includes the optimization elements (quantities, constraints and objectives) formulated in LP or MILP :

- Quantity : related to the decision variable or parameter
- Constraint : related to the optimization problem constraints
- Objective : related to the objective function

Copyright 2018 G2Elab / MAGE

Licensed under the Apache License, Version 2.0 (the “License”); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

```
class omegalpes.general.optimisation.elements.Constraint (exp,      name='CST0',
                                                         description=",      ac-
                                                         tive=True,      par-
                                                         ent=None)
```

Bases: object

### Description

Class that defines a constraint object

### Attributes

- name: name of the constraint
- description: a description of the constraint
- active: False = non-active constraint; True = active constraint
- exp: (str) : expression of the constraint
- parent: (unit) : this constraint belongs to this unit

---

**Note:** Make sure that all the modifications on Constraints are made before adding the unit to the Model (OptimisationModel.addUnit()).

---

```
class omegalpes.general.optimisation.elements.DynamicConstraint (exp_t,  
                                                             t_range='for  
                                                             t in time.I',  
                                                             name='DCST0',  
                                                             descrip-  
                                                             tion='dynamic  
                                                             constraint',  
                                                             active=True,  
                                                             parent=None)
```

Bases: *omegalpes.general.optimisation.elements.Constraint*

**\*\* Description \*\***

Defining a constraint depending on the time. NB : Mandatory for PuLP

```
class omegalpes.general.optimisation.elements.ExtDynConstraint (exp_t,  
                                                             t_range='for  
                                                             t in time.I',  
                                                             name='EDCST0',  
                                                             active=True,  
                                                             description='Non-  
                                                             physical and  
                                                             dynamic con-  
                                                             straint', par-  
                                                             ent=None)
```

Bases: *omegalpes.general.optimisation.elements.DynamicConstraint*, *omegalpes.general.optimisation.elements.ExternalConstraint*

**\*\* Description \*\***

Defining a constraint both external and dynamic (see: *DynamicConstraint*, *ExternalConstraint*)

```
class omegalpes.general.optimisation.elements.ExternalConstraint (exp,  
                                                             name='ExCST0',  
                                                             descrip-  
                                                             tion='',  
                                                             active=True,  
                                                             par-  
                                                             ent=None)
```

Bases: *omegalpes.general.optimisation.elements.Constraint*

**\*\* Description \*\***

**Defining a special type of constraint** [the external constraint]

- This constraint does not translate a physical constraint
- This constraint defines an external constraint, which could be relaxed

**deactivate\_constraint()**

An external constraint can be deactivated : - To compare scenarios - To try a less constrained problem

```
class omegalpes.general.optimisation.elements.HourlyDynamicConstraint (exp_t,
                                                                    time,
                                                                    init_h:
                                                                    int =
                                                                    0, fi-
                                                                    nal_h:
                                                                    int =
                                                                    24,
                                                                    name='HDCST0',
                                                                    de-
                                                                    scrip-
                                                                    tion='hourly
                                                                    dy-
                                                                    nam-
                                                                    ic
                                                                    con-
                                                                    straint',
                                                                    ac-
                                                                    tive=True,
                                                                    par-
                                                                    ent=None)
```

Bases: *omegalpes.general.optimisation.elements.DynamicConstraint*

**\*\* Description \*\***

Class that defines an dynamic contraint for a time range

Ex : Constraint applying between 7am and 10pm

ex\_cst = HourlyDynamicConstraint(exp\_t, time, init\_h=7, final\_h=22, name='ex\_cst')

#### Attributes

- name (str) : name of the constraint
- exp\_t (str) : expression of the constraint
- init\_h (int) : hour of beginning of the constraint [0-23]
- final\_h (int) : hour of end of the constraint [1-24]
- description (str) : description of the constraint
- active (bool) : defines if the constraint is active or not
- parent (Unit) : parent of the constraint

```
class omegalpes.general.optimisation.elements.Objective (exp, name='OBJ0', de-
                                                                    scription=", active=True,
                                                                    weight=1, unit='s.u.', par-
                                                                    ent=None)
```

Bases: object

#### Description

Class that defines an optimisation objective

#### Attributes

- name (str) :
- description (str) :
- active (bool) :

- `exp (str)` :
- `weight (float)` : weighted factor of the objective
- `parent (unit)`
- `unit (str)` : unit of the cost expression

---

**Note:** Make sure that all the modifications on Objectives are made before adding the unit to the Optimisation-Model, otherwise, it won't be taken into account

---

```
class omegalpes.general.optimisation.elements.Quantity (name='var0',    opt=True,  
unit='s.u',      vlen=None,  
value=None, description='',  
vtype='Continuous',  
lb=None,  ub=None,  parent=None)
```

Bases: `object`

### ” Description

Class that defines what is a quantity. A quantity can wether be a decision variable or a parameter, depending on the `opt` parameter

### Attributes

- `name (str)` : the name of the quantity
- `description (str)` : a description of the meaning of the quantity
- **`vtype (PuLP)`** [the variable type, depending on PuLP :]
  - `LpBinary` (binary variable)
  - `LpInteger` (integer variable)
  - `LpContinuous` (continuous variable)
- `vlen (int)` : size of the variable
- `unit (str)` : unit of the quantity
- **`opt (binary)`** :
  - `True`: this is an optimization variable
  - `False`: this is a constant - a parameter
- `value (float, list, dict)` : value (unused if `opt=True`)
- `ub, lb` : upper and lower bounds
- `parent (Unit)` : the quantity belongs to this unit

---

**Note:** Make sure that all the modifications on Quantity are made before adding the unit to the Model

---

## Model module

Copyright 2018 G2Elab / MAGE

Licensed under the Apache License, Version 2.0 (the “License”); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

```
class omegalpes.general.optimisation.model.OptimisationModel (time,
                                                             name='optimisation_model')
    Bases: pulp.pulp.LpProblem
    ** Description **
        This module includes the optimization model formulated in LP or MILP based on the package PuLP
        (LpProblem)
    add_nodes (*nodes)
        Add nodes and all connected units to the model Check that the time is the same for the model and all the
        units
        Parameters nodes – EnergyNode
    add_nodes_and_actors (*nodes_or_actors)
        Add nodes, actors and all connected units to the model Check that the time is the same for the model and
        all the units
        Parameters nodes_or_actors – EnergyNode or Actor type
    get_model_constraints_list ()
        Get constraints of the model
    get_model_constraints_name_list ()
        Get the names of the constraints of the model
    get_model_objectives_list ()
        Get objectives of the model
    get_model_objectives_name_list ()
        Get the names of the objectives of the model
    get_model_quantities_list ()
        Get quantities of the model
    get_model_quantities_name_list ()
        Get the names of the quantities of the model
    solve_and_update (solver: pulp.solvers.LpSolver = None) → None
        Solves the optimization model and updates all variables values.
        Parameters solver (LpSolver) – Optimization solver
    update_units ()
        Updates all units values with optimization results
    omegalpes.general.optimisation.model.check_if_unit_could_have_parent (unit)
        Check if the unit has an associated parent
        :param : unit
    omegalpes.general.optimisation.model.compute_gurobi_IIS (gurobi_exe_path='C:\\gurobi800\\win64\\bin',
                                                             opt_model=None,
                                                             MPS_model=None)
```

**Parameters**

- **gurobi\_exe\_path** – Path to the gurobi solver “gurobi\_cl.exe”
- **opt\_model** – OptimisationModel to whom compute IIS
- **MPS\_model** – name of the mps model

## Units module

” Copyright 2018 G2Elab / MAGE

Licensed under the Apache License, Version 2.0 (the “License”); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

```
class omegalpes.general.optimisation.units.Unit (name='U0', description='General unit')
```

Bases: object

### Description

Unit class is used as an “abstract class”, i.e. it defines some general attributes and methods but doesn’t contain variable, constraint nor objective. In the OMEGAAlpes package, all the subsystem models are represented by a unit. A model is then generated adding Unit to it. Variable, objective and constraints declarations are usually done using the `__init__` method of the Unit class.

### Attributes

- name
- description

### Methods

- `__str__`: defines the
- `__repr__`: defines the unit with its name
- `add_unit_attributes_in_lists`
- `get_constraints_list`
- `get_constraints_name_list`
- `get_objectives_list`
- `get_objectives_name_list`
- `get_quantities_list`
- `get_quantities_name_list`

---

**Note:** The Unit class shouldn’t be instantiated in a python script, except if you want to create your own model from the beginning. In this case, one should consider creating a new class `NewModel(Unit)`.

---

**add\_unit\_attributes\_in\_lists** () → None

#### Adds :

- The name of the Quantity elements contained in the unit to the list of quantities `self.quantities_list`



- The name of the Constraint elements contained in the unit to the list of constraints  
self.constraints\_list
- The name of the Objective elements contained in the unit to the list of objectives  
self.objectives\_list

**get\_constraints\_list()**

Get the constraints associated with the unit

**get\_constraints\_name\_list()**

Get the names of the constraints associated with the unit

**get\_objectives\_list()**

Get objectives associated with the unit

**get\_objectives\_name\_list()**

Get the names of the objectives associated with the unit

**get\_quantities\_list()**

Get the quantities associated with the unit

**get\_quantities\_name\_list()**

Get the names of the quantities associated with the unit

## Plots module

This module includes the display tools

Copyright 2018 G2Elab / MAGE

Licensed under the Apache License, Version 2.0 (the “License”); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

omegalpes.general.plots.**plot\_energy\_mix**(node)

omegalpes.general.plots.**plot\_node\_energetic\_flows**(node)

**\*\* Description \*\*** This function allows to plot the energy flows through an EnergyNode

**The display is realized :**

- with histograms for production and storage flow
- with dashed curve for consumption flow

**Parameters** node – EnergyNode

omegalpes.general.plots.**plot\_quantity**(time, q, fig=None, ax=None, color=None)

**Description**

Function that plots a OMEGALPES.general.optimisation.elements.Quantity

**Attributes**

- q is the Quantity
- fig could be None, a matplotlib.pyplot.Figure or Axes for multiple plots

**Returns**

- arg1 the matplotlib.pyplot.Figure handle object
- arg2 the matplotlib.pyplot.Axes handle object
- arg3 the matplotlib.pyplot.Line2D handle object

**Time module**

Copyright 2018 G2Elab / MAGE

Licensed under the Apache License, Version 2.0 (the “License”); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

**class** omegalpes.general.time.**TimeUnit** (*start='01/01/2018', end=None, periods=None, dt=1*)  
Bases: object

**Description**

Class defining the studied time period.

**Attributes**

- DATES : dated list of simulation steps
- DT : delta t between values in hours (int or float), i.e. 1/6 will be 10 minutes.
- LEN : number of simulation steps (length of DATES)
- I : index of time ([0 : LEN])

**get\_date\_for\_index** (*index*)

Getting a date for a given index

**Parameters** *index* – int value for the index of the wanted dated, between 0 and LEN (it must be in the studied period)

**get\_days**

Getting days for the studied period

**Return all\_days** list of days of the studied period

**get\_index\_for\_date** (*date='YYYY-MM-DD HH:MM:SS'*)

Getting the index associated with a date

**Parameters** *date* – date the index of is wanted. Format YYYY-MM-DD HH:MM:SS, must be within the studied period and consistent with the timestep value

**get\_index\_for\_date\_range** (*starting\_date='YYYY-MM-DD HH:MM:SS', end=None, periods=None*)

Getting a list of index for a date range

**Parameters**

- **starting\_date** – starting date of the wanted index
- **end** – ending date of the wanted index

- **periods** – number of periods from the starting\_date of the wanted index

**Return index\_list** list of indexes for the given dates

```
get_non_working_dates (month_range=range(0, 12), hour_range=range(0, 24), country='France')
```

```
get_non_working_days (country='France')
```

```
get_working_dates (month_range=range(0, 12), hour_range=range(0, 24), country='France')
```

```
get_working_days (country='France')
```

```
print_studied_period ()
```

```
omegalpes.general.time.convert_european_format (date)
```

Converting a date with an european format DD/MM/YYYY into a datetime format YYYY-MM-DD or return

**Parameters** **date** – date in european format

**Returns** date in format datetime

## Utils module

### Description

**This module includes the utils**

- To save results
- To define easily absolute value for quantity

Copyright 2018 G2Elab / MAGE

Licensed under the Apache License, Version 2.0 (the “License”); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

```
omegalpes.general.utils.def_abs_value (quantity, q_min, q_max)
```

**Parameters**

- **quantity** – Quantity whose absolute value is wanted
- **q\_min** – Minimal value of the quantity (negative value)
- **q\_max** – Maximal value of the quantity (positive value)

**Returns** A new Quantity whose values equal absolute values of the initial Quantity

```
omegalpes.general.utils.save_energy_flows (*nodes, file_name=None, sep='\n', decimal_sep='')
```

```
omegalpes.general.utils.select_csv_file_between_dates (file_path=None,  
                                                         start='DD/MM/YYYY  
                                                         HH:MM',  
                                                         end='DD/MM/YYYY  
                                                         HH:MM',          v_cols=[],  
                                                         sep=';')
```

## 2.5 OMEGAAlpes Examples

Please find OMEGAAlpes examples on the following link: <https://omegalpes-examples.readthedocs.io/>

It gathers detailed examples to explain the steps in order to create a model and article study cases.

## CHAPTER 3

---

### Acknowledgments

---

Vincent Reinbold - Library For Linear Modeling of Energetic Systems : <https://github.com/ReinboldV>

Mathieu Brugeron

This work has been partially supported by the [CDP Eco-SESA](#) receiving fund from the French National Research Agency in the framework of the “Investissements d’avenir” program (ANR-15-IDEX-02) and the VALocal project (CNRS Interdisciplinary Mission and INSIS)



### O

`omegalpes.actor.actor`, [30](#)  
`omegalpes.actor.operator_actors.consumer_actors`,  
    [31](#)  
`omegalpes.actor.operator_actors.consumer_producer_actors`,  
    [36](#)  
`omegalpes.actor.operator_actors.operator_actors`,  
    [31](#)  
`omegalpes.actor.operator_actors.producer_actors`,  
    [33](#)  
`omegalpes.actor.regulator_actors.regulator_actors`,  
    [38](#)  
`omegalpes.energy.energy_nodes`, [27](#)  
`omegalpes.energy.io.poles`, [29](#)  
`omegalpes.energy.units.consumption_units`,  
    [11](#)  
`omegalpes.energy.units.conversion_units`,  
    [23](#)  
`omegalpes.energy.units.energy_units`, [7](#)  
`omegalpes.energy.units.production_units`,  
    [17](#)  
`omegalpes.energy.units.storage_units`,  
    [25](#)  
`omegalpes.general.optimisation.elements`,  
    [39](#)  
`omegalpes.general.optimisation.model`,  
    [42](#)  
`omegalpes.general.optimisation.units`,  
    [44](#)  
`omegalpes.general.plots`, [45](#)  
`omegalpes.general.time`, [46](#)  
`omegalpes.general.utils`, [47](#)





## A

Actor (class in *omegalpes.actor.actor*), 30

add\_availability()  
(*omegalpes.energy.units.energy\_units.EnergyUnit*  
method), 8

add\_co2\_emissions()  
(*omegalpes.energy.units.energy\_units.EnergyUnit*  
method), 8

add\_connected\_energy\_unit()  
(*omegalpes.energy.energy\_nodes.EnergyNode*  
method), 28

add\_external\_constraint()  
(*omegalpes.actor.actor.Actor* method), 30

add\_external\_dynamic\_constraint()  
(*omegalpes.actor.actor.Actor* method), 30

add\_max\_ramp\_down()  
(*omegalpes.energy.units.energy\_units.EnergyUnit*  
method), 8

add\_max\_ramp\_up()  
(*omegalpes.energy.units.energy\_units.EnergyUnit*  
method), 8

add\_min\_time\_off()  
(*omegalpes.energy.units.energy\_units.EnergyUnit*  
method), 8

add\_min\_time\_on()  
(*omegalpes.energy.units.energy\_units.EnergyUnit*  
method), 8

add\_nodes() (*omegalpes.general.optimisation.model.OptimisationModel*  
method), 43

add\_nodes\_and\_actors()  
(*omegalpes.general.optimisation.model.OptimisationModel*  
method), 43

add\_objective() (*omegalpes.actor.actor.Actor*  
method), 30

add\_operating\_cost()  
(*omegalpes.energy.units.energy\_units.EnergyUnit*  
method), 8

add\_pole() (*omegalpes.energy.energy\_nodes.EnergyNode*  
method), 28

add\_starting\_cost()  
(*omegalpes.energy.units.energy\_units.EnergyUnit*  
method), 8

add\_unit\_attributes\_in\_lists()  
(*omegalpes.general.optimisation.units.Unit*  
method), 44

## C

check\_if\_unit\_could\_have\_parent() (in  
module *omegalpes.general.optimisation.model*),  
43

co2\_emission\_maximum()  
(*omegalpes.actor.regulator\_actors.regulator\_actors.RegulatorAct*  
method), 38

compute\_gurobi\_IIS() (in module  
*omegalpes.general.optimisation.model*), 43

connect\_units() (*omegalpes.energy.energy\_nodes.EnergyNode*  
method), 28

Constraint (class in  
*omegalpes.general.optimisation.elements*),  
39

Consumer (class in *omegalpes.actor.operator\_actors.consumer\_actors*),  
32

ConsumptionUnit (class in  
*omegalpes.energy.units.consumption\_units*), 12

ConversionUnit (class in  
*omegalpes.energy.units.conversion\_units*),  
14

## M

convert\_european\_format() (in module  
*omegalpes.general.time*), 47

export() (*omegalpes.energy.energy\_nodes.EnergyNode*  
method), 28

## D

deactivate\_constraint()  
(*omegalpes.general.optimisation.elements.ExternalConstraint*  
method), 40

def\_abs\_value() (in module  
*omegalpes.general.utils*), 47

DynamicConstraint (class in *omegalpes.general.optimisation.elements*), 39

## E

ElectricalToHeatConversionUnit (class in *omegalpes.energy.units.conversion\_units*), 24

energy\_consumption\_maximum() (*omegalpes.actor.operator\_actors.consumer\_actors.Consumer* method), 32

energy\_consumption\_minimum() (*omegalpes.actor.operator\_actors.consumer\_actors.Consumer* method), 32

energy\_production\_maximum() (*omegalpes.actor.operator\_actors.producer\_actors.Producer* method), 34

energy\_production\_minimum() (*omegalpes.actor.operator\_actors.producer\_actors.Producer* method), 34

EnergyNode (class in *omegalpes.energy.energy\_nodes*), 27

EnergyUnit (class in *omegalpes.energy.units.energy\_units*), 7

Epole (class in *omegalpes.energy.io.poles*), 29

export\_to\_node() (*omegalpes.energy.energy\_nodes.EnergyNode* method), 28

ExtDynConstraint (class in *omegalpes.general.optimisation.elements*), 40

ExternalConstraint (class in *omegalpes.general.optimisation.elements*), 40

## F

FixedConsumptionUnit (class in *omegalpes.energy.units.consumption\_units*), 13

FixedEnergyUnit (class in *omegalpes.energy.units.energy\_units*), 9

FixedProductionUnit (class in *omegalpes.energy.units.production\_units*), 18

FlowPole (class in *omegalpes.energy.io.poles*), 29

## G

get\_connected\_energy\_units (*omegalpes.energy.energy\_nodes.EnergyNode* attribute), 28

get\_constraints\_list() (*omegalpes.general.optimisation.units.Unit* method), 45

get\_constraints\_name\_list() (*omegalpes.general.optimisation.units.Unit* method), 45

get\_date\_for\_index() (*omegalpes.general.time.TimeUnit* method), 46

get\_days (*omegalpes.general.time.TimeUnit* attribute), 46

get\_exports (*omegalpes.energy.energy\_nodes.EnergyNode* attribute), 28

get\_flows (*omegalpes.energy.energy\_nodes.EnergyNode* attribute), 28

get\_imports (*omegalpes.energy.energy\_nodes.EnergyNode* attribute), 28

get\_index\_for\_date() (*omegalpes.general.time.TimeUnit* method), 46

get\_index\_for\_date\_range() (*omegalpes.general.time.TimeUnit* method), 46

get\_model\_constraints\_list() (*omegalpes.general.optimisation.model.OptimisationModel* method), 43

get\_model\_constraints\_name\_list() (*omegalpes.general.optimisation.model.OptimisationModel* method), 43

get\_model\_objectives\_list() (*omegalpes.general.optimisation.model.OptimisationModel* method), 43

get\_model\_objectives\_name\_list() (*omegalpes.general.optimisation.model.OptimisationModel* method), 43

get\_model\_quantities\_list() (*omegalpes.general.optimisation.model.OptimisationModel* method), 43

get\_model\_quantities\_name\_list() (*omegalpes.general.optimisation.model.OptimisationModel* method), 43

get\_non\_working\_dates() (*omegalpes.general.time.TimeUnit* method), 47

get\_non\_working\_days() (*omegalpes.general.time.TimeUnit* method), 47

get\_objectives\_list() (*omegalpes.general.optimisation.units.Unit* method), 45

get\_objectives\_name\_list() (*omegalpes.general.optimisation.units.Unit* method), 45

get\_poles (*omegalpes.energy.energy\_nodes.EnergyNode* attribute), 28

get\_quantities\_list() (*omegalpes.general.optimisation.units.Unit* method), 45

get\_quantities\_name\_list() (*omegalpes.general.optimisation.units.Unit* method), 45

get\_working\_dates() (*omegalpes.general.time.TimeUnit* method), 47

get\_working\_days() (*omegalpes.general.time.TimeUnit* method), 47

## H

HeatPump (class in *omegalpes.energy.units.conversion\_units*),  
24  
HourlyDynamicConstraint (class in *omegalpes.general.optimisation.elements*),  
40

## I

import\_from\_node()  
(*omegalpes.energy.energy\_nodes.EnergyNode*  
method), 28  
is\_export\_flow() (*omegalpes.energy.energy\_nodes.EnergyNode*  
method), 28  
is\_import\_flow() (*omegalpes.energy.energy\_nodes.EnergyNode*  
method), 29

## L

LocalAuthorities (class in *omegalpes.actor.regulator\_actors.regulator\_actors*),  
38

## M

maximize\_conso\_prod\_match()  
(*omegalpes.actor.operator\_actors.consumer\_producer\_actors.Producer*  
method), 36  
maximize\_consumption()  
(*omegalpes.actor.operator\_actors.consumer\_actors.Consumer*  
method), 32  
maximize\_consumption()  
(*omegalpes.energy.units.consumption\_units.ConsumptionUnit*  
method), 13  
maximize\_production()  
(*omegalpes.actor.operator\_actors.producer\_actors.Producer*  
method), 34  
maximize\_production()  
(*omegalpes.energy.units.production\_units.ProductionUnit*  
method), 19  
maximize\_selfconsumption\_rate()  
(*omegalpes.actor.operator\_actors.consumer\_producer\_actors.Producer*  
method), 36  
maximize\_selfproduction\_rate()  
(*omegalpes.actor.operator\_actors.consumer\_producer\_actors.Producer*  
method), 37  
minimize\_capacity()  
(*omegalpes.energy.units.storage\_units.StorageUnit*  
method), 27  
minimize\_co2\_consumption()  
(*omegalpes.actor.operator\_actors.consumer\_actors.Consumer*  
method), 32  
minimize\_co2\_emissions()  
(*omegalpes.actor.operator\_actors.producer\_actors.Producer*  
method), 34

## O

Objective (class in  
*omegalpes.general.optimisation.elements*),  
41  
*omegalpes.actor.actor* (module), 30  
*omegalpes.actor.operator\_actors.consumer\_actors*  
(module), 31

omegalpes.actor.operator\_actors.consumer\_producer\_period() (module), 36  
 omegalpes.actor.operator\_actors.operator\_producer (class in omegalpes.actor.operator\_actors.producer\_actors), (module), 31  
 omegalpes.actor.operator\_actors.producer\_producer\_unit (class in omegalpes.energy.units.production\_units), (module), 33  
 omegalpes.actor.regulator\_actors.regulator\_actors (module), 38  
 omegalpes.energy.energy\_nodes (module), 27  
 omegalpes.energy.io.poles (module), 29  
 omegalpes.energy.units.consumption\_units (module), 11  
 omegalpes.energy.units.conversion\_units (module), 23  
 omegalpes.energy.units.energy\_units (module), 7  
 omegalpes.energy.units.production\_units (module), 17  
 omegalpes.energy.units.storage\_units (module), 25  
 omegalpes.general.optimisation.elements (module), 39  
 omegalpes.general.optimisation.model (module), 42  
 omegalpes.general.optimisation.units (module), 44  
 omegalpes.general.plots (module), 45  
 omegalpes.general.time (module), 46  
 omegalpes.general.utils (module), 47  
 OperatorActor (class in omegalpes.actor.operator\_actors.operator\_actors), 31  
 OptimisationModel (class in omegalpes.general.optimisation.model), 43  
**P**  
 plot\_energy\_mix() (in module omegalpes.general.plots), 45  
 plot\_node\_energetic\_flows() (in module omegalpes.general.plots), 45  
 plot\_quantity() (in module omegalpes.general.plots), 45  
 power\_consumption\_maximum() (omegalpes.actor.operator\_actors.consumer\_actors.Consumer method), 33  
 power\_consumption\_minimum() (omegalpes.actor.operator\_actors.consumer\_actors.Consumer method), 33  
 power\_production\_maximum() (omegalpes.actor.operator\_actors.producer\_actors.Producer method), 35  
 power\_production\_minimum() (omegalpes.actor.operator\_actors.producer\_actors.Producer method), 35  
 print\_standard\_period() (omegalpes.general.time.TimeUnit method), 47  
 Producer (class in omegalpes.actor.operator\_actors.producer\_actors), 34  
 ProducerUnit (class in omegalpes.energy.units.production\_units), 19  
 Prosumer (class in omegalpes.actor.operator\_actors.consumer\_producer\_actors), 36  
 PublicAuthorities (class in omegalpes.actor.regulator\_actors.regulator\_actors), 38  
**Q**  
 Quantity (class in omegalpes.general.optimisation.elements), 42  
**R**  
 RegulatorActor (class in omegalpes.actor.regulator\_actors.regulator\_actors), 38  
**S**  
 save\_energy\_flows() (in module omegalpes.general.utils), 47  
 SawtoothEnergyUnit (class in omegalpes.energy.units.energy\_units), 9  
 select\_csv\_file\_between\_dates() (in module omegalpes.general.utils), 47  
 set\_energy\_limits\_on\_time\_period() (omegalpes.energy.units.energy\_units.EnergyUnit method), 9  
 set\_power\_balance() (omegalpes.energy.energy\_nodes.EnergyNode method), 29  
 SeveralConsumptionUnit (class in omegalpes.energy.units.consumption\_units), 13  
 SeveralEnergyUnit (class in omegalpes.energy.units.energy\_units), 9  
 SeveralImaginaryConsumptionUnit (class in omegalpes.energy.units.consumption\_units), 14  
 SeveralImaginaryProductionUnit (class in omegalpes.energy.units.production\_units), 19  
 SeveralProductionUnit (class in omegalpes.energy.units.production\_units), 20  
 ShiftableConsumptionUnit (class in omegalpes.energy.units.consumption\_units), 15  
 ShiftableEnergyUnit (class in omegalpes.energy.units.energy\_units), 10  
 ShiftableProductionUnit (class in omegalpes.energy.units.production\_units), 21

`solve_and_update()`  
     (*omegalpes.general.optimisation.model.OptimisationModel*  
     *method*), 43

`SquareConsumptionUnit` (class in  
     *omegalpes.energy.units.consumption\_units*), 16

`SquareEnergyUnit` (class in  
     *omegalpes.energy.units.energy\_units*), 11

`SquareProductionUnit` (class in  
     *omegalpes.energy.units.production\_units*),  
     22

`StorageUnit` (class in  
     *omegalpes.energy.units.storage\_units*), 25

`Supplier` (class in *omegalpes.actor.operator\_actors.consumer\_producer\_actors*),  
     37

## T

`ThermoclineStorage` (class in  
     *omegalpes.energy.units.storage\_units*), 27

`TimeUnit` (class in *omegalpes.general.time*), 46

`TriangleEnergyUnit` (class in  
     *omegalpes.energy.units.energy\_units*), 11

## U

`Unit` (class in *omegalpes.general.optimisation.units*), 44

`update_units()` (*omegalpes.general.optimisation.model.OptimisationModel*  
     *method*), 43

## V

`VariableConsumptionUnit` (class in  
     *omegalpes.energy.units.consumption\_units*), 17

`VariableEnergyUnit` (class in  
     *omegalpes.energy.units.energy\_units*), 11

`VariableProductionUnit` (class in  
     *omegalpes.energy.units.production\_units*),  
     22