

---

**optimeed**

***Release 2.0.2***

**Aug 07, 2022**



---

## Contents

---

<b>1</b>	<b>Requirements</b>	<b>3</b>
<b>2</b>	<b>Installation</b>	<b>5</b>
<b>3</b>	<b>Quickstart</b>	<b>7</b>
3.1	Quickstart Optimization . . . . .	7
3.2	Quickstart Visualization . . . . .	10
3.3	Loading and saving data . . . . .	12
<b>4</b>	<b>Gallery</b>	<b>15</b>
4.1	Gallery . . . . .	15
<b>5</b>	<b>License and support</b>	<b>17</b>
5.1	License and Support . . . . .	17
<b>6</b>	<b>API</b>	<b>19</b>
6.1	<code>:py:mod:optimeed</code> . . . . .	19
<b>7</b>	<b>Developer guide</b>	<b>159</b>
7.1	Developer documentation . . . . .	159
	<b>Python Module Index</b>	<b>161</b>
	<b>Index</b>	<b>163</b>



Optimeed is a free open source package that allows to perform optimization and data visualization/management.



# CHAPTER 1

---

## Requirements

---

- PyQt5 for visualisation -> `pip install PyQt5`
- *pyopengl* for visualisation -> `pip install PyOpenGL`
- Numpy -> `pip install numpy`
- **Optional**
  - pandas which is only used to export excel files -> `pip install pandas`
  - nlopt library for using other types of algorithm. -> `pip install nlopt`
  - inkscape software for exporting graphs in .png and .pdf)





## CHAPTER 2

---

### Installation

---

To install the latest optimeed release, run the following command:

```
pip install optimeed
```

To install the latest development version of optimeed, run the following commands:

```
git clone https://git.immc.ucl.ac.be/chdegreef/optimeed.git
cd optimeed
python setup.py install
```



Examples can be found [on the tutorial folder](#) .

### 3.1 Quickstart Optimization

An optimization process can be presented as following:

- **Optimization algorithm:** *AlgorithmInterface*. This is the algorithm that performs the optimization, and outputs a vector of variables between  $[0, 1[$ .
- **Maths to physics:** *InterfaceMathsToPhysics*. Transforms the output vector of the optimization algorithm to the variables of a *InterfaceDevice*. The usage of this block becomes meaningful for more complex optimization problem, such as optimizing a BLDC motor while keeping the outer diameter constant. In this case, a good implementation of the M2P block automatically scales the inner dimensions of the motor to comply with this constraint.
- **Characterization:** *InterfaceCharacterization*. Based on the attributes of the device, performs some computation. This block is nearly useless for simple optimization problems (when the objective function is easily computed) but becomes interesting for more complex problems, where many things need to be precalculated before obtaining the objective functions and constraints. This for example can hold an analytical or a FEM magnetic model. A sub-optimization could also be performed there.
- **Objective and constraints:** *InterfaceObjCons*. These classes correspond to either what has to be minimized, or which constraints  $\leq 0$  has to be complied with.

Quick example:  $\min_{x,y \in [0,2]} f(x) = \sqrt{1 + (y + 3) \cdot x^2}, g(x) = 4 + 2\sqrt{y + 3} \cdot \sqrt{1 + (x - 1)^2}$ , under the constrained that  $x \leq 0.55$ . This is a bi-objective problem and will lead to a pareto front.

```
"""Now that you know everything about data and visualization, let's get started with
↳ optimization!
Optimpeed provides high-level interface to perform optimization with visualization and
↳ data storage.
The Wiki gives more details about the optimization. To get started, you need the
↳ following key ingredients:
```

(continues on next page)

(continued from previous page)

```

- A device that contains the variables to be optimized ("Device") and other_
↳ parameters you would like to save
- A list of optimization variables ("OptimizationVariable")
- An evaluation function ("Characterization")
- One or more objective functions ("Objectives")
- (optional) Eventual constraints ("Constraints")
- An optimization algorithm ("Optimization Algorithm")
- Something that will fill the "Device" object with the optimization variables_
↳ coming from the optimization algorithm. ("MathsToPhysics")
    Don't get scared with this one, if you do not know how it can be useful, the_
↳ proposition by default works perfectly fine.
- Something that will link all the blocks together ("Optimizer")
"""

# These are what we need for the optimization
from optimeed.optimize.optiAlgorithms import MultiObjective_GA as_
↳ OptimizationAlgorithm
from optimeed.optimize import Real_OptimizationVariable, InterfaceObjCons,_
↳ InterfaceCharacterization, OptiHistoric
from optimeed.optimize.optimizer import OptimizerSettings, run_optimization

# These are the high-level visualization tools
from optimeed.visualize.displayOptimization import OptimizationDisplayer
from optimeed.visualize import Onclick_representDevice, Represent_brut_attributes,_
↳ start_qt_mainloop
import time

class Device:
    """Define the Device to optimize."""
    x: float # Type hinted -> will be automatically saved
    y: float # Type hinted -> will be automatically saved

    def __init__(self):
        self.x = 1
        self.y = 1

class Characterization(InterfaceCharacterization):
    """Define the Characterization scheme. In this case nothing is performed,
    but this is typically where model code will be executed and results saved inside
    ↳ 'theDevice'."""
    def compute(self, thedevice):
        time.sleep(0.0001)

class MyObjective1(InterfaceObjCons):
    """First objective function (to be minimized)"""
    def compute(self, thedevice):
        return (1 + (thedevice.y+3)*thedevice.x**2)**0.5

class MyObjective2(InterfaceObjCons):
    """Second objective function (to be minimized)"""
    def compute(self, thedevice):

```

(continues on next page)

(continued from previous page)

```

        return 4 + 2*(thedevice.y+3)**0.5*(1+(thedevice.x-1)**2)**0.5

class MyConstraint (InterfaceObjCons):
    """Constraints, that needs to be <= 0"""
    def compute(self, thedevice):
        return thedevice.x - 0.55

if __name__ == "__main__": # This line is necessary to spawn new processes
    """Start the main code. Instantiate previously defined classes."""
    theDevice = Device()
    theAlgo = OptimizationAlgorithm()
    theAlgo.set_option(theAlgo.OPTI_ALGORITHM, "NSGAI") # You can change the
    ↪algorithm if you need ;)
    # theAlgo.set_option(theAlgo.NUMBER_OF_CORES, 2) # Toggle this line to use more
    ↪cores. Default is 1 (single core).

    # Careful that it generates overhead -> only helpful when the characterization is
    ↪computationally expensive.

    theCharacterization = Characterization()

    """Variable to be optimized"""
    optimizationVariables = list()
    optimizationVariables.append(Real_OptimizationVariable('x', 0, 2)) #
    optimizationVariables.append(Real_OptimizationVariable('y', 0, 2))

    """Objective and constraints"""
    listOfObjectives = [MyObjective1(), MyObjective2()]
    listOfConstraints = [MyConstraint()]

    """Set the optimizer"""
    theOptiParameters = OptimizerSettings(theDevice, listOfObjectives,
    ↪listOfConstraints, optimizationVariables,
                                theOptimizationAlgorithm=theAlgo,
    ↪theCharacterization=theCharacterization)

    """The logger (to automatically save the points)"""
    theOptiHistoric = OptiHistoric(optiname="opti", autosave_timer=10, autosave=True,
    ↪create_new_directory=True)

    """Start the optimization"""
    max_opti_time_sec = 10

    display_opti = True
    if display_opti: # Display real-time graphs
        optiDisplayer = OptimizationDisplayer(theOptiParameters, theOptiHistoric,
    ↪light_background=True)
        _, theDataLink, _ = optiDisplayer.generate_optimizationGraphs()

        # Here we set the actions on click.
        theActionsOnClick = list()
        theActionsOnClick.append(OnClick_representDevice(theDataLink, [Represent_brut_
    ↪attributes()])))
        optiDisplayer.set_actionsOnClick(theActionsOnClick)

```

(continues on next page)

(continued from previous page)

```

        resultsOpti, convergence = optiDisplayer.launch_
→ optimization([theOptiParameters, theOptiHistoric], {"max_opti_time_sec": max_opti_
→ time_sec},
                                                    refresh_time=0.1,
→ max_nb_points_convergence=None) # Refresh the graphs each nth seconds

    else: # Otherwise just focus on results ... That can be helpful if you are_
→ confident the optimizations will converge and you need to launch several_
→ optimizations.
        resultsOpti, convergence = run_optimization(theOptiParameters,
→ theOptiHistoric, max_opti_time_sec=max_opti_time_sec)

    """Gather results"""
    # Pro hint: you would probably never work with these next few lines of code,
→ instead you would move to the next tutorial
    # to retrieve the results from the automatically saved files.
    print("Best individuals :")
    for device in resultsOpti:
        print("x : {} \t y : {}".format(device.x, device.y))

    if display_opti:
        start_qt_mainloop() # To keep windows alive

    """Note that the results are automatically saved if KWARGS_OPTIHISTO_
→ autosaved=True.
    In this case, optimization folder is automatically generated in Workspace/optiX.
→ It contains five files:
    -> autosaved: contains all the devices evaluated during the optimization
    -> logopti: contains all the information relating to the optimization itself:
→ objectives, constraints, evaluation time.
    -> opticonvergence: contains all the information relative to the convergence of
→ the optimization (saved only at the end)
    -> results: all the best devices as decided by the optimization algorithm
    -> optimization_parameters: the class OptimizationParameters that can be reloaded
→ using SingleObjectSaveLoad.load
    -> summary.html: a summary of the optimization problem
    See other tutorials on how to save/load these information.
    """

```

## 3.2 Quickstart Visualization

Visualization implies to have a GUI, which will help to display many things: graphs, text, 3D representations, ... This software provides a clean interface to PyQt. PyQt works that way:

- A QMainWindow that includes layouts, (ex: horizontal, vertical, grid, ...)
- Layouts can include widgets.
- Widgets can be anything: buttons, menu, opengl 3D representation, graphs, ... Several high-level widgets are proposed, check `optimeed.visualize.widgets`.

### 3.2.1 Simple gui using OpenGL:

```

"""Optimeed also provides visualization tools using openGL. Here's how.
"""

# We already know these imports ...
from optimeed.core import ListDataStruct
from optimeed.core import LinkDataGraph, HowToPlotGraph
from optimeed.visualize import Widget_graphsVisual, MainWindow

# Now for the openGL imports:
from optimeed.visualize.widgets import Widget_openGL # And we put all of that inside_
↳ a widget
from optimeed.visualize.openGL import DeviceDrawerInterface, Bronze_material
from optimeed.visualize.openGL.openGL_library import glPushMatrix, draw_simple_
↳ rectangle, glPopMatrix, glTranslate
from optimeed.visualize.onclick import Onclick_animate, Animate_openGL # And we put_
↳ the widget inside an action

class MyDevice:
    def __init__(self, length, height):
        self.length = length
        self.height = height
        self.surface = self.length * self.height

# The drawer class
class DeviceDrawer(DeviceDrawerInterface):
    """Drawer of the device"""
    def __init__(self):
        self.theDevice = None

    def draw(self, theDevice):
        glPushMatrix() # Remove the previous matrices transformations
        glTranslate(-5, -5, 0)
        Bronze_material.activateMaterialProperties() # Change colour aspect of the_
↳ material, here it will look like bronze
        draw_simple_rectangle(theDevice.length, theDevice.height) # Thats the_
↳ interesting line
        glPopMatrix() # Push back previous matrices transformations

    def get_init_camera(self, theDevice):
        tipAngle = 0
        viewAngle = 0
        zoomLevel = 0.1
        return tipAngle, viewAngle, zoomLevel

# From now on, that's the same as before
theDataStruct = ListDataStruct()
for xi in range(10):
    for yi in range(10):
        theDataStruct.add_data(MyDevice(xi, yi))

theDataLink = LinkDataGraph()
_ = theDataLink.add_collection(theDataStruct)

```

(continues on next page)

(continued from previous page)

```

howToPlot = HowToPlotGraph('length', 'surface', {'x_label': "length [m]", 'y_label':
↳ "surface [m^2]", 'is_scattered': True})
theDataLink.add_graph(howToPlot)
theGraphs = theDataLink.get_graphs()

theActionsOnClick = list()

# That's where the fun begins
openGLDrawing = Widget_openGL() # First we define the widget
openGLDrawing.set_deviceDrawer(DeviceDrawer()) # We set the drawer to the widget
theActionsOnClick.append(OnClick_animate(theDataLink, Animate_openGL(openGLDrawing)))
↳ # And we create an action out of it

# Same as before
myWidgetGraphsVisuals = Widget_graphsVisual(theGraphs,
↳ actionsOnClick=theActionsOnClick, highlight_last=True, refresh_time=-1) # The_
↳ widget to display the graphs
myWindow = MainWindow([myWidgetGraphsVisuals]) # A Window (that will contain the_
↳ widget)
myWindow.run(True)

# Now click on a point.
# Click on "show all"
# Watch the graph and animation together

```

### 3.3 Loading and saving data

You will probably have to often manipulate data, saving them and loading them.

Imagine the following structure to be saved:

```

class TopoA:
    def __init__(self):
        self.R_in = 3e-3
        self.R_out = 5e-3

class MyMotor:
    def __init__(self):
        self.rotor = TopoA()
        self.length = 5e-3
        self.dummyVariableToNotSave = 1234

```

optimeed provides a way to export that directly in JSON format. It detects the variables to save from type hints:

```

class TopoA:
    R_in: float
    R_out: float

    def __init__(self):
        self.R_in = 3e-3
        self.R_out = 5e-3

```

(continues on next page)



(continued from previous page)

```

class MyMotor:
    rotor: TopoA
    length: float

    def __init__(self):
        self.rotor = TopoA()
        self.length = 5e-3
        self.dummyVariableToNotSave = 1234

```

If type hint is not possible because some type is not known before the running time, optimeed provides an additional tool *SaveableObject*:

```

from optimeed.core import SaveableObject

class TopoA:
    R_in: float
    R_out: float

    def __init__(self):
        self.R_in = 3e-3
        self.R_out = 5e-3

class MyMotor(SaveableObject):
    length: float

    def __init__(self):
        self.rotor = TopoA()
        self.length = 5e-3
        self.dummyVariableToNotSave = 1234

    def get_additional_attributes_to_save(self):
        return ["rotor"]

```

The item can then be converted to a dictionary using *obj\_to\_json()*, which can then be converted to string liberal using “json.dumps” and written on a file. To recover To recover the object, read the file and interpret is as a dictionary using “json.load”. Then, convert the dictionary by using *json\_to\_obj()*

Alternatively, it might be simpler to use the class *ListDataStruct* (or similar user-custom class), which provides high-level save and load option. This is what is done in *OptiHistoric*



## CHAPTER 4

---

### Gallery

---

#### 4.1 Gallery



## 5.1 License and Support

### 5.1.1 License

The project is distributed “has it is” under [GNU General Public License v3.0 \(GPL\)](#), which is a strong copyleft license. This means that the code is open-source and you are free to do anything you want with it, **as long as you apply the same license to distribute your code**. This constraining license is imposed by the use of [Platypus Library](#) as “optimization algorithm library”, which is under GPL license.

It is perfectly possible to use other optimization library (which would use the same algorithms but with a different implementation) and to interface it to this project, so that the use of platypus is no longer needed. This work has already been done for [NLOpt](#), which is under MIT license (not constraining at all). In that case, **after removing all the platypus sources** (`optiAlgorithms/multiObjective_GA` and `optiAlgorithms/platypus/*`), the license of the present work becomes less restrictive: [GNU Lesser General Public License \(LGPL\)](#). As for the GPL, this license makes the project open-source and free to be modified, but (nearly) no limitation is made to distribute your code.

### 5.1.2 Support

Github (preferably) / Send mail at [christophe.degreef@uclouvain.be](mailto:christophe.degreef@uclouvain.be)



## 6.1 :py:mod:optimeed

### 6.1.1 Subpackages

**consolidate**

**fit**

#### Module Contents

#### Classes

#### Functions

**class** `_Device` (*fitFunction, nbArgs*)

**class** `_Objective` (*x\_data, y\_data, fitCriterion*)

Bases: `optimeed.optimize.InterfaceObjCons`

Interface class for objectives and constraints. The objective is to MINIMIZE and the constraint has to respect  $VALUE \leq 0$

**compute** (*theDevice*)

Get the value of the objective or the constraint. The objective is to MINIMIZE and the constraint has to respect  $VALUE \leq 0$

**Parameters** *theDevice* – Input device that has already been evaluated

**Returns** float.

**leastSquare** (*function, functionArgs, x\_data, y\_data*)

Least square calculation ( $\sum (y-\hat{y})^2$ )

**Parameters**

- **function** – Function to fit
- **functionArgs** – Arguments of the function
- **x\_data** – x-axis coordinates of data to fit
- **y\_data** – y-axis coordinates of data to fit

**Returns** least squares

**r\_squared** (*function, functionArgs, x\_data, y\_data*)

R squared calculation

**Parameters**

- **function** – Function to fit
- **functionArgs** – Arguments of the function
- **x\_data** – x-axis coordinates of data to fit
- **y\_data** – y-axis coordinates of data to fit

**Returns** R squared

**do\_fit** (*fitFunction, x\_data, y\_data, \*args, fitCriterion=leastSquare*)

Main method to fit a function

**Parameters**

- **fitFunction** – the function to fit (link to it)
- **x\_data** – x-axis coordinates of data to fit
- **y\_data** – y-axis coordinates of data to fit
- **args** – for each parameter: [min, max] admissible value
- **fitCriterion** – fit criterion to minimize. Default: least square

**Returns** [arg\_i\_optimal, ...], y estimated, error.

**parametric\_analysis****Module Contents****Classes**

**class Parametric\_Collection** (*\*\*kwargs*)

Bases: *optimeed.core.collection.ListDataStruct*

**class Parametric\_parameter** (*analyzed\_attribute, reference\_device*)

Abstract class for a parametric parameter

**get\_reference\_device** ()

**get\_analyzed\_attribute** ()

**class Parametric\_minmax** (*analyzed\_attribute, reference\_device, minValue, maxValue, is\_relative=False, npoints=10*)

Bases: *Parametric\_parameter*

Abstract class for a parametric parameter



```

    get_values ()
class Parametric_analysis (theParametricParameter,          theCharacterization,          file-
                           name_collection=None, autosave=False)
    Bases: optimeed.core.Option_class
    NUMBER_OF_CORES = 1
    run ()
        Instantiates input arguments for analysis
    evaluate (theDevice)
    initialize_output_collection ()

```

## sensitivity\_analysis

### Module Contents

#### Classes

#### Functions

#### Attributes

```

_filename_sensitivityparams = sensitivity_params.json
_foldername_embarrassingly_parallel_results = _jobs_results
_filename_sensitivityresults = sensitivity.json

```

#### class SensitivityResults

Bases: *optimeed.core.SaveableObject*

Abstract class for dynamically type-hinted objects. This class is to solve the special case where the exact type of an attribute is not known before runtime, yet has to be saved.

**paramsToEvaluate** :List[float]

**success** :bool

**index** :int

**add\_data** (params, device, success, index)

**get\_additional\_attributes\_to\_save** ()

Return list of attributes corresponding to object, whose type cannot be determined statically (e.g. topology change)

#### class SensitivityParameters (param\_values, list\_of\_optimization\_variables, theDevice, theMath- sToPhys, theCharacterization)

Bases: *optimeed.core.SaveableObject*

Abstract class for dynamically type-hinted objects. This class is to solve the special case where the exact type of an attribute is not known before runtime, yet has to be saved.

**list\_of\_optimization\_variables** :List[optimeed.optimize.Real\_OptimizationVariable]

**param\_values** :List[List[float]]

**get\_device** ()

**get\_M2P** ()

**get\_charac** ()

**get\_optivariab**les ()

**get\_paramvalues** ()

**get\_additional\_attributes\_to\_save** ()

Return list of attributes corresponding to object, whose type cannot be determined statically (e.g. topology change)

**get\_sensitivity\_problem** (*list\_of\_optimization\_variables*)

This is the first method to use. Convert a list of optimization variables to a SALib problem

**Parameters** *list\_of\_optimization\_variables* – List of optimization variables

**Returns** SALib problem

**\_get\_sensitivity\_result** (*output*)

Convert output of “evaluate” function to SensitivityResult

**\_get\_job\_args** (*theSensitivityParameters, index*)

Convert sensitivityparameters at index to args used in “evaluate” function

**\_find\_missings** (*theSensitivityParameters, studyn*ame)

**prepare\_embarrassingly\_parallel\_sensitivity** (*theSensitivityParameters, studyn*ame)

**launch\_embarrassingly\_parallel\_sensitivity** (*theSensitivityParameters, studyn*ame, *index*)

**gather\_embarrassingly\_parallel\_sensitivity** (*theSensitivityParameters, studyn*ame)

**evaluate\_sensitivities** (*theSensitivityParameters: SensitivityParameters, numberOfCores=2, studyn*ame=*'sensitivity'*, *indices\_to\_evaluate=None*)

Evaluate the sensitivities

**Parameters**

- **theSensitivityParameters** – class:~SensitivityParameters
- **numberOfCores** – number of core for multicore evaluation
- **studyn**ame – Name of the study, that will be the subfolder name in workspace
- **indices\_to\_evaluate** – if None, evaluate all param\_values, otherwise if list: evaluate subset of param\_values defined by indices\_to\_evaluate

**Returns** collection of class:~SensitivityResults

**analyse\_sobol\_create\_array** (*theSensitivityParameters: SensitivityParameters, objectives*)

Create readable result array, ordered by decreasing sobol indices.

**Parameters**

- **theSensitivityParameters** – class:SensitivityParameters
- **objectives** – array-like of objective

**Returns** tuples of STR, for S1 and ST

**analyse\_sobol\_convergence** (*theSensitivityParameters: SensitivityParameters, objectives, step-size=1*)

Create dictionary for convergence plot

**Parameters**

- **theSensitivityParameters** – class:SensitivityParameters

- **objectives** – array-like of objective

**Returns** Dictionary

**sensitivity\_analysis\_evaluation**

## Module Contents

### Functions

**evaluate** (*inputs*)

## Package Contents

### Classes

### Functions

**class Option\_class**

```

options_bool :Dict[int, Option_bool]
options_str :Dict[int, Option_str]
options_int :Dict[int, Option_int]
options_float :Dict[int, Option_float]
options_dict :Dict[int, Option_dict]
add_option (idOption, theOption)
get_option_name (idOption)
get_option_value (idOption)
set_option (idOption, value)
_pack_options ()
__str__ ()
    Return str(self).
```

**class Option\_int** (*name, based\_value, choices=None*)

Bases: Base\_Option

**name** :str

**value** :int

**set\_value** (*value*)

**class ListDataStruct** (*compress\_save=False*)

Bases: ListDataStruct\_Interface

**\_DATA\_STR** = data

**\_COMPRESS\_SAVE\_STR** = module\_tree

**\_\_len\_\_** ()

**get\_length()**

**clone** (*filename*)

Clone the datastructure to a new location

**save** (*filename*)

Save data using json format. The data to be saved are automatically detected, see *obj\_to\_json()*

**extract\_collection\_from\_indices** (*indices*)

Extract data from the collection at specific indices, and return it as new collection

**\_format\_str\_save** ()

Save data using json format. The data to be saved are automatically detected, see *obj\_to\_json()*

**\_format\_data\_lines** ()

**\_get\_json\_module\_tree** ()

**add\_data** (*data\_in*)

Add a data to the list

**get\_data** ()

Get full list of datas

**get\_data\_generator** ()

Get a generator to all the data stored

**get\_data\_at\_index** (*index*)

**set\_data** (*theData*)

Set full list of datas

**set\_data\_at\_index** (*data\_in*, *index*)

Replace data at specific index

**reset\_data** ()

**delete\_points\_at\_indices** (*indices*)

Delete several elements from the Collection

**Parameters** *indices* – list of indices to delete

**merge** (*collection*)

Merge a collection with the current collection

**Parameters** *collection* – Collection to merge

**get\_nbr\_elements** ()

**Returns** the number of elements contained inside the structure

**class AutosaveStruct** (*dataStruct*, *filename*="", *change\_filename\_if\_exists*=True)

Structure that provides automated save of DataStructures

**\_\_str\_\_** ()

Return str(self).

**get\_filename** ()

Get set filename

**set\_filename** (*filename*, *change\_filename\_if\_exists*)

**Parameters**

- **filename** – Filename to set
- **change\_filename\_if\_exists** – If already exists, create a new filename

```

stop_autosave ()
    Stop autosave

start_autosave (timer_autosave, safe_save=True)
    Start autosave

save (safe_save=True)
    Save

get_datastruct ()
    Return :class:'~DataStruct_Interface'

__getstate__ ()

__setstate__ (state)

getPath_workspace ()
    Get workspace path (i.e., location where optimeed files will be created). Create directory if doesn't exist.

rsetattr (obj, attr, val)
    setattr, but recursively. Works with list (i.e. theObj.myList[0].var_x)

rgetattr (obj, attr)
    getattr, but recursively. Works with list.

class Parametric_Collection (**kwargs)
    Bases: optimeed.core.collection.ListDataStruct

class Parametric_parameter (analyzed_attribute, reference_device)
    Abstract class for a parametric parameter

    get_reference_device ()

    get_analyzed_attribute ()

class Parametric_minmax (analyzed_attribute, reference_device, minValue, maxValue,
                           is_relative=False, npoints=10)
    Bases: Parametric_parameter

    Abstract class for a parametric parameter

    get_values ()

class Parametric_analysis (theParametricParameter, theCharacterization, file-
                           name_collection=None, autosave=False)
    Bases: optimeed.core.Option_class

    NUMBER_OF_CORES = 1

    run ()
        Instantiates input arguments for analysis

    evaluate (theDevice)

    initialize_output_collection ()

leastSquare (function, functionArgs, x_data, y_data)
    Least square calculation (sum (y-ŷ)^2)

    Parameters
    • function – Function to fit
    • functionArgs – Arguments of the function
    • x_data – x-axis coordinates of data to fit

```

- **y\_data** – y-axis coordinates of data to fit

**Returns** least squares

**do\_fit** (*fitFunction, x\_data, y\_data, \*args, fitCriterion=leastSquare*)

Main method to fit a function

**Parameters**

- **fitFunction** – the function to fit (link to it)
- **x\_data** – x-axis coordinates of data to fit
- **y\_data** – y-axis coordinates of data to fit
- **args** – for each parameter: [min, max] admissible value
- **fitCriterion** – fit criterion to minimize. Default: least square

**Returns** [arg\_i\_optimal, ...], y estimated, error.

**get\_sensitivity\_problem** (*list\_of\_optimization\_variables*)

This is the first method to use. Convert a list of optimization variables to a SALib problem

**Parameters** **list\_of\_optimization\_variables** – List of optimization variables

**Returns** SALib problem

**evaluate\_sensitivities** (*theSensitivityParameters: SensitivityParameters, numberOfCores=2, study-name='sensitivity', indices\_to\_evaluate=None*)

Evaluate the sensitivities

**Parameters**

- **theSensitivityParameters** – class '~SensitivityParameters'
- **numberOfCores** – number of core for multicore evaluation
- **studyname** – Name of the study, that will be the subfolder name in workspace
- **indices\_to\_evaluate** – if None, evaluate all param\_values, otherwise if list: evaluate subset of param\_values defined by indices\_to\_evaluate

**Returns** collection of class '~SensitivityResults'

**class SensitivityParameters** (*param\_values, list\_of\_optimization\_variables, theDevice, theMathsToPhys, theCharacterization*)

Bases: *optimeed.core.SaveableObject*

Abstract class for dynamically type-hinted objects. This class is to solve the special case where the exact type of an attribute is not known before runtime, yet has to be saved.

**list\_of\_optimization\_variables** :List[optimeed.optimize.Real\_OptimizationVariable]

**param\_values** :List[List[float]]

**get\_device** ()

**get\_M2P** ()

**get\_charac** ()

**get\_optivariabes** ()

**get\_paramvalues** ()

**get\_additional\_attributes\_to\_save** ()

Return list of attributes corresponding to object, whose type cannot be determined statically (e.g. topology change)

**analyse\_sobol\_convergence** (*theSensitivityParameters*: *SensitivityParameters*, *objectives*, *step-size=1*)  
Create dictionary for convergence plot

#### Parameters

- **theSensitivityParameters** – class:*SensitivityParameters*
- **objectives** – array-like of objective

**Returns** Dictionary

**prepare\_embarrassingly\_parallel\_sensitivity** (*theSensitivityParameters*, *studyname*)

**gather\_embarrassingly\_parallel\_sensitivity** (*theSensitivityParameters*, *studyname*)

**launch\_embarrassingly\_parallel\_sensitivity** (*theSensitivityParameters*, *studyname*, *index*)

core

## Subpackages

ansi2html

converter

## Module Contents

### Classes

### Functions

### Attributes

ANSI\_FULL\_RESET = 0

ANSI\_INTENSITY\_INCREASED = 1

ANSI\_INTENSITY\_REDUCED = 2

ANSI\_INTENSITY\_NORMAL = 22

ANSI\_STYLE\_ITALIC = 3

ANSI\_STYLE\_NORMAL = 23

ANSI\_BLINK\_SLOW = 5

ANSI\_BLINK\_FAST = 6

ANSI\_BLINK\_OFF = 25

ANSI\_UNDERLINE\_ON = 4

ANSI\_UNDERLINE\_OFF = 24

ANSI\_CROSSED\_OUT\_ON = 9

ANSI\_CROSSED\_OUT\_OFF = 29

ANSI\_VISIBILITY\_ON = 28

```
ANSI_VISIBILITY_OFF = 8
ANSI_FOREGROUND_CUSTOM_MIN = 30
ANSI_FOREGROUND_CUSTOM_MAX = 37
ANSI_FOREGROUND_256 = 38
ANSI_FOREGROUND_DEFAULT = 39
ANSI_BACKGROUND_CUSTOM_MIN = 40
ANSI_BACKGROUND_CUSTOM_MAX = 47
ANSI_BACKGROUND_256 = 48
ANSI_BACKGROUND_DEFAULT = 49
ANSI_NEGATIVE_ON = 7
ANSI_NEGATIVE_OFF = 27
ANSI_FOREGROUND_HIGH_INTENSITY_MIN = 90
ANSI_FOREGROUND_HIGH_INTENSITY_MAX = 97
ANSI_BACKGROUND_HIGH_INTENSITY_MIN = 100
ANSI_BACKGROUND_HIGH_INTENSITY_MAX = 107
VT100_BOX_CODES
```

```
_latex_template = Multiline-String
```

```
1 \documentclass{scrartcl}
2 \usepackage[utf8]{inputenc}
3 \usepackage{fancyvrb}
4 \usepackage[usenames,dvipsnames]{xcolor}
5 %% \definecolor{red-sd}{HTML}{7ed2d2}
6
7 \title{%(title)s}
8
9 \fvset{commandchars=\\\{\}}
10
11 \begin{document}
12
13 \begin{Verbatim}
14 %(content)s
15 \end{Verbatim}
16 \end{document}
```

```
_html_template
```

```
class _State
```

```
    Bases: object
```

```
    reset ()
```

```
    adjust (ansi_code, parameter=None)
```

```
    to_css_classes ()
```

```
linkify (line, latex_mode)
```

```
map_vt100_box_code (char)
```

```
_needs_extra_newline (text)
```



```
class CursorMoveUp
```

```
    Bases: object
```

```
class Ansi2HTMLConverter(latex=False, inline=False, dark_bg=True, line_wrap=True,  
                        font_size='normal', linkify=False, escaped=True, markup_lines=False,  
                        output_encoding='utf-8', scheme='ansi2html', title='')
```

```
    Bases: object
```

```
    Convert Ansi color codes to CSS+HTML
```

```
    Example: >>> conv = Ansi2HTMLConverter() >>> ansi = ""
    conv.convert(ansi)
```

```
    apply_regex (ansi)
```

```
    _apply_regex (ansi, styles_used)
```

```
    _collapse_cursor (parts)
```

```
        Act on any CursorMoveUp commands by deleting preceding tokens
```

```
    prepare (ansi="", ensure_trailing_newline=False)
```

```
        Load the contents of 'ansi' into this object
```

```
    attrs ()
```

```
        Prepare attributes for the template
```

```
    convert (ansi, full=True, ensure_trailing_newline=False)
```

```
    produce_headers ()
```

```
main ()
```

```
    $ ls -color=always | ansi2html > directories.html $ sudo tail /var/log/messages | ccze -A | ansi2html > logs.html
```

```
    $ task burndown | ansi2html > burndown.html
```

**style**

## Module Contents

### Classes

### Functions

### Attributes

```
class Rule(klass, **kw)
```

```
    Bases: object
```

```
    __str__ ()
```

```
        Return str(self).
```

```
index (r, g, b)
```

```
color_component (x)
```

```
color (r, g, b)
```

```
level (grey)
```

```
index2 (grey)
```

```
SCHEME
```

```
intensify (color, dark_bg, amount=64)  
get_styles (dark_bg=True, line_wrap=True, scheme='ansi2html')
```

**util**

## Module Contents

### Functions

```
read_to_unicode (obj)
```

### Package Contents

### Classes

```
class Ansi2HTMLConverter (latex=False, inline=False, dark_bg=True, line_wrap=True,  
                           font_size='normal', linkify=False, escaped=True, markup_lines=False,  
                           output_encoding='utf-8', scheme='ansi2html', title='')
```

Bases: object

Convert Ansi color codes to CSS+HTML

Example: >>> conv = Ansi2HTMLConverter() >>> ansi = "" “.join(sys.stdin.readlines()) >>> html = conv.convert(ansi)

```
apply_regex (ansi)
```

```
_apply_regex (ansi, styles_used)
```

```
_collapse_cursor (parts)
```

Act on any CursorMoveUp commands by deleting preceding tokens

```
prepare (ansi="", ensure_trailing_newline=False)
```

Load the contents of 'ansi' into this object

```
attrs ()
```

Prepare attributes for the template

```
convert (ansi, full=True, ensure_trailing_newline=False)
```

```
produce_headers ()
```

**additional\_tools**

## Module Contents

### Classes

### Functions

### Attributes

```
has_scipy = True
```

---

**class fast\_LUT\_interpolation** (*independent\_variables, dependent\_variables*)  
 Class designed for fast interpolation in look-up table when successive searches are called often. Otherwise use griddata

**interpolate** (*point, fill\_value=np.nan*)  
 Perform the interpolation :param point: coordinates to interpolate (tuple or list of tuples for multipoints)  
 :param fill\_value: value to put if extrapolated. :return: coordinates

**interpolate\_table** (*x0, x\_values, y\_values*)  
 From sorted table (x,y) find y0 corresponding to x0 (linear interpolation)

**derivate** (*t, y*)

**linspace** (*start, stop, npoints*)

**reconstitute\_signal** (*amplitudes, phases, numberOfPeriods=1, x\_points=None, n\_points=50*)  
 Reconstitute the signal from fft. Number of periods of the signal must be specified if different of 1

**my\_fft** (*y*)  
 Real FFT of signal Bx, with real amplitude of harmonics. Input signal must be within a period.

**cart2pol** (*x, y*)

**pol2cart** (*rho, phi*)

**partition** (*array, begin, end*)

**quicksort** (*array*)

**dist** (*p, q*)  
 Return the Euclidean distance between points p and q. :param p: [x, y] :param q: [x, y] :return: distance (float)

**sparse\_subset** (*points, r*)  
 Returns a maximal list of elements of points such that no pairs of points in the result have distance less than r.  
 :param points: list of tuples (x,y) :param r: distance :return: corresponding subset (list), indices of the subset (list)

**integrate** (*x, y*)  
 Performs Integral(x[0] to x[-1]) of y dx

**Parameters**

- **x** – x axis coordinates (list)
- **y** – y axis coordinates (list)

**Returns** integral value

**my\_fourier** (*x, y, n, L*)  
 Fourier analys

**Parameters**

- **x** – x axis coordinates
- **y** – y axis coordinates
- **n** – number of considered harmonic
- **L** – half-period length

**Returns** a and b coefficients ( $y = a*\cos(x) + b*\sin(y)$ )

**get\_ellipse\_axes** (*a, b, dphi*)  
 Trouve les longueurs des axes majeurs et mineurs de l'ellipse, ainsi que l'orientation de l'ellipse. ellipse:  $x(t) =$

$A \cos(t), y(t) = B \cos(t + \phi)$  Etapes: longueur demi ellipse CENTRÉE =  $\sqrt{a^2 \cos^2(x) + b^2 \cos^2(t + \phi)}$   
Minimisation de cette formule => obtention formule  $\tan(2x) = \alpha/\beta$

**convert\_color** (*color*)

Convert a color to a tuple if color is a char, otherwise return the tuple.

**Parameters** **color** – (r,g,b) or char.

**Returns**

**convert\_color\_with\_alpha** (*color*, *alpha*=255)

Same as meth:*convert\_color* but with transparency

**collection**

## Module Contents

### Classes

### Attributes

**class** SingleObjectSaveLoad

**class** DataStruct\_Interface

\_\_str\_\_ ()

Return str(self).

**class** ListDataStruct\_Interface

Bases: *DataStruct\_Interface*

**get\_list\_attributes** (*attributeName*)

Get the value of attributeName of all the data in the Collection

**Parameters** **attributeName** – string (name of the attribute to get)

**Returns** list

**class** AutosaveStruct (*dataStruct*, *filename*=", *change\_filename\_if\_exists*=True)

Structure that provides automated save of DataStructures

\_\_str\_\_ ()

Return str(self).

**get\_filename** ()

Get set filename

**set\_filename** (*filename*, *change\_filename\_if\_exists*)

**Parameters**

- **filename** – Filename to set
- **change\_filename\_if\_exists** – If already exists, create a new filename

**stop\_autosave** ()

Stop autosave

**start\_autosave** (*timer\_autosave*, *safe\_save*=True)

Start autosave

```

save (safe_save=True)
    Save

get_datastruct ()
    Return :class:`~DataStruct_Interface`

__getstate__ ()

__setstate__ (state)

class ListDataStruct (compress_save=False)
    Bases: ListDataStruct_Interface

    _DATA_STR = data

    _COMPRESS_SAVE_STR = module_tree

    __len__ ()

    get_length ()

    clone (filename)
        Clone the datastructure to a new location

    save (filename)
        Save data using json format. The data to be saved are automatically detected, see obj_to_json()

    extract_collection_from_indices (indices)
        Extract data from the collection at specific indices, and return it as new collection

    _format_str_save ()
        Save data using json format. The data to be saved are automatically detected, see obj_to_json()

    _format_data_lines ()

    _get_json_module_tree ()

    add_data (data_in)
        Add a data to the list

    get_data ()
        Get full list of datas

    get_data_generator ()
        Get a generator to all the data stored

    get_data_at_index (index)

    set_data (theData)
        Set full list of datas

    set_data_at_index (data_in, index)
        Replace data at specific index

    reset_data ()

    delete_points_at_indices (indices)
        Delete several elements from the Collection

        Parameters indices – list of indices to delete

    merge (collection)
        Merge a collection with the current collection

        Parameters collection – Collection to merge

    get_nbr_elements ()

```

**Returns** the number of elements contained inside the structure

**theLock**

**class Performance\_ListDataStruct** (*stack\_size=500*)

Bases: *ListDataStruct\_Interface*

**\_NBR\_ELEMENTS** = *nbr\_elements*

**\_STACK\_SIZE** = *stack\_size*

**\_COMPRESS\_SAVE\_STR** = *module\_tree*

**\_initialize** (*filename*)

**\_get\_list\_from\_file** (*filenumber*)

**extract\_collection\_from\_indices** (*indices*)

Extract data from the collection at specific indices, and return it as new collection

**clone** (*filename*)

Clone the datastructure to a new location

**\_get\_str\_mainfile** ()

**get\_total\_nbr\_elements** (*count\_unsaved=True*)

**add\_data** (*theData*)

Add data to the collection

**add\_json\_data** (*theStr*)

Add already deserialized data to the collection

**\_save\_moduletree** (*theDict*)

**\_map\_index\_to\_file** (*index*)

**\_get\_json\_str\_at\_index** (*index, refresh\_cache=False*)

Internal method to return the json string at index

**reorder** (*permutations*)

Reorder collection accordingly to permutations. E.G, list\_of\_indices = [0,3,2] with collection elems [0,2,1]  
=> collection elems = [0,2,3] :param permutations: :return: /

**get\_data\_at\_index** (*index, ignore\_attributes=None, none\_if\_error=False*)

Same as parent, with additional kwargs

#### Parameters

- **index** –
- **ignore\_attributes** – ignore attributes to deserialize (list)
- **none\_if\_error** –

#### Returns

**save** (*filename*)

Save the datastructure to filename

**get\_data\_generator** (*\*\*kwargs*)

**get\_nbr\_elements** ()

**Returns** the number of elements contained inside the structure

**set\_data\_at\_index** (*data\_in, index*)

Replace data at specific index

**set\_data\_at\_indices** (*data\_list, indices*)

Replace datas at specific indices :param data\_list: list of objects to set to the collection, at specific indices  
:param indices: list of indices :return:

**delete\_points\_at\_indices** (*indices*)

Delete several elements from the Collection

**Parameters indices** – list of indices to delete

`color_palette`

## Module Contents

### Functions

**default\_palette** (*N*)

**blackOnly** (*N*)

**dark2** (*N*)

`commonImport`

## Module Contents

### Functions

### Attributes

**SHOW\_WARNING** = 0

**SHOW\_INFO** = 1

**SHOW\_ERROR** = 2

**SHOW\_DEBUG** = 3

**SHOW\_LOGS** = 4

**SHOW\_CURRENT**

**setCurrentShow** (*show\_types*)

Change text type to be displayed by PrintfShown

**getCurrentShow** ()

Get text type to be displayed by PrintfShown

**disableLogs** ()

Disable all logs

**enableLogs** ()

Show all logs

## graphs

### Module Contents

#### Classes

**class Data** (*x: list, y: list, x\_label="", y\_label="", legend="", is\_scattered=False, transfo\_x=lambda self-Data, x: x, transfo\_y=lambda selfData, y: y, xlim=None, ylim=None, permutations=None, sort\_output=False, color=None, alpha=255, symbol='o', symbolsize=8, fillsymbol=True, out-linesymbol=1.8, linestyle='-', width=2, meta=None*)

This class is used to store informations necessary to plot a 2D graph. It has to be combined with a gui to be useful (ex. pyqtgraph)

**set\_kwargs** (*kwargs*)

Set a kwarg after creation of the class

**set\_data** (*x: list, y: list*)

Overwrites current datapoints with new set

**set\_meta** (*meta*)

Set associated 'Z' data

**get\_x** ()

Get x coordinates of datapoints

**get\_symbolsize** ()

Get size of the symbols

**symbol\_isfilled** ()

Check if symbols has to be filled or not

**get\_symbolOutline** ()

Get color factor of outline of symbols

**get\_length\_data** ()

Get number of points

**get\_xlim** ()

Get x limits of viewbox

**get\_ylim** ()

Get y limits of viewbox

**get\_y** ()

Get y coordinates of datapoints

**get\_meta** ()

Get associated 'Z' data

**get\_color** ()

Get color of the line, without transformation

**get\_color\_alpha** ()

Get color of the line. Return r, g, b in 0, 255 scale

**get\_alpha** ()

Get opacity

**get\_width** ()

Get width of the line



**get\_number\_of\_points()**

Get number of points

**get\_plot\_data()**

Call this method to get the x and y coordinates of the points that have to be displayed. => After transformation, and after permutations.

**Returns** x (list), y (list)

**get\_plot\_meta(x, y)**

Call this method to get the z coordinates of the points that been displayed. => After transformation, and after permutations.

**Returns** z (list)

**get\_permutations(x=None)**

Return the transformation 'permutation': xplot[i] = xdata[permutation[i]]

**get\_invert\_permutations()**

Return the inverse of permutations: xdata[i] = xplot[revert[i]]

**get\_dataIndex\_from\_graphIndex(index\_graph\_point)**

From an index given in graph, recovers the index of the data.

**Parameters** **index\_graph\_point** – Index in the graph

**Returns** index of the data

**get\_dataIndices\_from\_graphIndices(index\_graph\_point\_list)**

Same as get\_dataIndex\_from\_graphIndex but with a list in entry. Can (?) improve performances for huge dataset.

**Parameters** **index\_graph\_point\_list** – List of Index in the graph

**Returns** List of index of the data

**get\_graphIndex\_from\_dataIndex(index\_data)**

From an index given in the data, recovers the index of the graph.

**Parameters** **index\_data** – Index in the data

**Returns** index of the graph

**get\_graphIndices\_from\_dataIndices(index\_data\_list)**

Same as get\_graphIndex\_from\_dataIndex but with a list in entry. Can (?) improve performances for huge dataset.

**Parameters** **index\_data\_list** – List of Index in the data

**Returns** List of index of the graph

**set\_permutations(permutations)**

Set permutations between datapoints of the trace

**Parameters** **permutations** – list of indices to plot (example: [0, 2, 1] means that the first point will be plotted, then the third, then the second one)

**get\_x\_label()**

Get x label of the trace

**get\_y\_label()**

Get y label of the trace

**get\_legend()**

Get name of the trace

**get\_symbol()**  
Get symbol

**add\_point(x, y)**  
Add point(s) to trace (inputs can be list or numeral)

**delete\_point(index\_point)**  
Delete a point from the datapoints

**isScattered()**  
Check if plot is scattered

**set\_indices\_points\_to\_plot(indices)**  
Set indices points to plot

**get\_indices\_points\_to\_plot()**  
Get indices points to plot

**get\_linestyle()**  
Get linestyle

**\_\_str\_\_()**  
Return str(self).

**export\_str()**  
Method to save the points constituting the trace

**set\_color(theColor)**  
Set trace color

**set\_legend(theLegend)**  
Set legend

**class Graph**

Simple graph container that contains several traces

**add\_trace(data)**  
Add a trace to the graph

**Parameters** *data* – *Data*

**Returns** id of the created trace

**remove\_trace(idTrace)**  
Delete a trace from the graph

**Parameters** *idTrace* – id of the trace to delete

**get\_trace(idTrace) → Data**  
Get data object of *idTrace*

**Parameters** *idTrace* – id of the trace to get

**Returns** *Data*

**get\_all\_traces()**  
Get all the traces id of the graph

**get\_all\_traces\_ids()**  
Get all the traces id of the graph :return: list of id graphs

**export\_str()**

**class Graphs**

Contains several *Graph*

**updateChildren** ()

**add\_trace\_firstGraph** (*data*, *updateChildren=True*)

Same as add\_trace, but only if graphs has only one id :param data: :param updateChildren: :return:

**add\_trace** (*idGraph*, *data*, *updateChildren=True*)

Add a trace to the graph

**Parameters**

- **idGraph** – id of the graph
- **data** – *Data*
- **updateChildren** – Automatically calls callback functions

**Returns** id of the created trace

**remove\_trace** (*idGraph*, *idTrace*, *updateChildren=True*)

Remove the trace from the graph

**Parameters**

- **idGraph** – id of the graph
- **idTrace** – id of the trace to remove
- **updateChildren** – Automatically calls callback functions

**get\_first\_graph** ()

Get id of the first graph

**Returns** id of the first graph

**get\_graph** (*idGraph*)

Get graph object at idgraph

**Parameters** **idGraph** – id of the graph to get

**Returns** *Graph*

**get\_all\_graphs\_ids** ()

Get all ids of the graphs

**Returns** list of id graphs

**get\_all\_graphs** ()

Get all graphs. Return dict {id: *Graph*}

**add\_graph** (*updateChildren=True*)

Add a new graph

**Returns** id of the created graph

**remove\_graph** (*idGraph*)

Delete a graph

**Parameters** **idGraph** – id of the graph to delete

**add\_update\_method** (*childObject*)

Add a callback each time a graph is modified.

**Parameters** **childObject** – method without arguments

**export\_str** ()

Export all the graphs in text

**Returns** str

```
merge (otherGraphs)
reset ()
is_empty ()
```

graphs3

## Module Contents

### Classes

### Functions

### Attributes

```
griddata_found = True
```

```
class Plot3D_Generic (x_label="", y_label="", z_label="", legend="", x_lim=None, y_lim=None,
                      z_lim=None)
```

```
    get_lim (axis)
    get_label (axis)
    get_legend ()
```

```
class GridPlot_Generic (X, Y, Z, **kwargs)
    Bases: Plot3D_Generic
    get_plot_data ()
```

```
class ContourPlot (*args, **kwargs)
    Bases: GridPlot_Generic
    get_levels ()
    get_number_of_contours ()
```

```
class FilledContourPlot (*args, **kwargs)
    Bases: ContourPlot
```

```
class SurfPlot (X, Y, Z, **kwargs)
    Bases: GridPlot_Generic
```

```
class MeshPlot (X, Y, Z, **kwargs)
    Bases: GridPlot_Generic
```

```
class ScatterPlot3 (x, y, z, **kwargs)
    Bases: Plot3D_Generic
    get_plot_data ()
    get_color ()
```

```
convert_to_gridplot (x, y, z, x_interval=None, y_interval=None, n_x=20, n_y=20)
    Convert set of points x, y, z to a grid
```

#### Parameters

- **x** –

- **y** –
- **z** –
- **x\_interval** – [Min, max] of the grid. If none, use min and max values
- **y\_interval** – [Min, max] of the grid. If none, use min and max values
- **n\_x** – number of points in x direction
- **n\_y** – number of points in y direction

**Returns** X, Y, Z as grid

`inkscape_manager`

## Module Contents

### Functions

### Attributes

`get_path_to_inkscape()`

`get_inkscape_version()`

`inkscape_version`

`inkscape_svg_to_pdf(filename_svg, filename_pdf)`

`inkscape_svg_to_png(filename_svg, filename_png)`

`linkDataGraph`

## Module Contents

### Classes

**class** `HowToPlotGraph` (*attribute\_x, attribute\_y, kwargs\_graph=None, check\_if\_plot\_elem=None, meta=None*)

`__str__()`  
Return str(self).

**class** `LinkDataGraph`

**add\_collection** (*theCollection, kwargs=None*)  
Add a collection (that will be a future trace)

#### Parameters

- **theCollection** –
- **kwargs** – kwargs associated with the collection (e.g., color, symbol style, etc.)

**Returns** unique id associated with the collection

**remove\_collection** (*collectionId*)

Remove collection from the graphs

**Parameters** *collectionId* – ID of the collection

**Returns**

**set\_shadow\_collection** (*master\_collectionId*, *shadow\_collection*)

Link a collection to an other

**Parameters**

- **master\_collectionId** – ID of the collection that is displayed in the graph
- **shadow\_collection** – collection to link to the master.

**Returns**

**get\_graphs** ()

**get\_howToPlotGraph** (*idGraph*)

**add\_graph** (*howToPlotGraph*)

Add new graph to be plotted.

**Parameters** *howToPlotGraph* – *HowToPlotGraph*

**Returns**

**get\_idCollections** ()

Get all ids of the plotted collections

**get\_idGraphs** ()

Get all ids of the graphs

**get\_idTraces** (*idGraph*)

Get all ids of the traces of graph \$idGraph

**get\_idCollection\_from\_graph** (*idGraph*, *idTrace*)

Get id of collection plotted in graph \$idGraph and trace \$idTrace

**get\_collection** (*idCollection*, *getShadow=True*)

**update\_graphs** ()

Update the graphs: update graphs, traces, and X-Y data

**get\_collection\_from\_graph** (*idGraph*, *idTrace*, *getShadow=True*) → opti-  
meed.core.ListDataStruct\_Interface

From indices in the graph, get corresponding collection

**get\_clicked\_item** (*idGraph*, *idTrace*, *idPoint*, *getShadow=True*)

Get the data hidden behind the clicked point

**Parameters**

- **idGraph** – ID of the graph
- **idTrace** – ID of the trace
- **idPoint** – ID of the point
- **getShadow** – If true, will return the data from the collection linked to the collection that is plotted

**Returns** Object in collection

**get\_clicked\_items** (*idGraph*, *idTrace*, *idPoint\_list*, *getShadow=True*)

Same as `get_clicked_item`, but using a list of points

**delete\_clicked\_item** (*idGraph*, *idTrace*, *idPoint*)  
 Remove item from the collection

**delete\_clicked\_items** (*idGraph*, *idTrace*, *idPoints*)  
 Same, but for a list of points

**get\_graph\_and\_trace\_from\_idCollection** (*idCollection*)  
 Reverse search: from a collection, get all associated graphs

**get\_idcollection\_from\_collection** (*theCollection*)  
 Reverse search: from a collection, find its id

**get\_idPoints\_from\_indices\_in\_collection** (*idGraph*, *idTrace*, *indices\_in\_collection*)  
 From indices in a collection, find the associated idPoints of the graph

myjson

## Module Contents

### Classes

### Functions

### Attributes

**MODULE\_TAG** = `__module__`

**CLASS\_TAG** = `__class__`

**EXCLUDED\_TAGS**

**getExecPath** ()

**class SaveableObject**

Abstract class for dynamically type-hinted objects. This class is to solve the special case where the exact type of an attribute is not known before runtime, yet has to be saved.

**get\_additional\_attributes\_to\_save** ()

Return list of attributes corresponding to object, whose type cannot be determined statically (e.g. topology change)

**get\_additional\_attributes\_to\_save\_list** ()

Same behavior as `get_additional_attributes_to_save`, but where the attributes contains list of unknown items

**\_\_isclass** (*theObject*)

Extends the default isclass method with typing

**get\_type\_class** (*typ*)

Get the type of the class. used to compare objects from Typing.

**\_\_get\_object\_class** (*theObj*)

**\_\_get\_object\_module** (*theObj*)

**\_\_object\_to\_FQCN** (*theobj*)

Gets module path of object

**\_\_find\_class** (*moduleName*, *className*)

**json\_to\_obj** (*json\_dict*)

Convenience class to create object from dictionary. Only works if CLASS\_TAG is valid

**Parameters** **json\_dict** – dictionary loaded from a json file.

**Raises**

- **TypeError** – if class can not be found
- **KeyError** – if CLASS\_TAG not present in dictionary

**json\_to\_obj\_safe** (*json\_dict, cls*)

Safe class to create object from dictionary.

**Parameters**

- **json\_dict** – dictionary loaded from a json file
- **cls** – class object to instantiate with dictionary

**\_instantiates\_annotated\_object** (*\_json\_dict, \_cls*)

**\_get\_annotations** (*theObj*)

Return annotated attributes (theObj being the type of the object)

**obj\_to\_json** (*theObj*)

Extract the json dictionary from the object. The data saved are automatically detected, using typehints. ex: x: int=5 will be saved, x=5 won't. Inheritance of annotation is managed by this function

**\_get\_attributes\_to\_save** (*theObj*)

Return list (attribute, is\_first)

**get\_json\_module\_tree\_from\_dict** (*jsonDict*)

Return dict containing {CLASS\_TAG: "class\_name", MODULE\_TAG: "module\_name", "attribute1":{"class\_name": "module\_name", ... }}

**remove\_module\_tree\_from\_string** (*theStr*)

Used to compress string by removing \_\_module\_\_ and \_\_class\_\_ entries (used with get\_json\_module\_tree\_from\_dict)

**apply\_module\_tree\_to\_dict** (*nestedTree, nestedObject, raiseError=False*)

Restore \_\_module\_\_ and \_\_class\_\_ entries from nestedTree in nestedDict

**encode\_str\_json** (*theStr*)

**decode\_str\_json** (*theStr*)

**options**

## Module Contents

### Classes

**class Base\_Option** (*name, based\_value, choices=None*)

**get\_value** ()

**get\_name** ()

**set\_value** (*value*)

**get\_choices** ()



```

class Option_bool(name, based_value, choices=None)
    Bases: Base_Option

    name :str
    value :bool
    set_value(value)
    get_choices()

class Option_str(name, based_value, choices=None)
    Bases: Base_Option

    name :str
    value :str
    set_value(value)

class Option_int(name, based_value, choices=None)
    Bases: Base_Option

    name :str
    value :int
    set_value(value)

class Option_float(name, based_value, choices=None)
    Bases: Base_Option

    name :str
    value :float
    set_value(value)

class Option_dict(name, based_value, choices=None)
    Bases: Base_Option

    name :str
    value :dict
    set_value(value)

class Option_class

    options_bool :Dict[int, Option_bool]
    options_str :Dict[int, Option_str]
    options_int :Dict[int, Option_int]
    options_float :Dict[int, Option_float]
    options_dict :Dict[int, Option_dict]
    add_option(idOption, theOption)
    get_option_name(idOption)
    get_option_value(idOption)
    set_option(idOption, value)
    _pack_options()

```

`__str__()`  
Return str(self).

`tikzTranslator`

## Module Contents

### Functions

### Attributes

`templates_tikz`

`format_escape_char(theStr)`

`convert_linestyle(linestyle)`

`find_all_colors(theGraphs)`

`convert_marker(marker)`

`do_preamble()`

`do_generate_figure()`

`do_specific_axis_options(theGraph: optimeed.core.graphs.Graph)`  
Get graph-specific axis options

`do_specific_trace_options(theTrace: optimeed.core.graphs.Data, theColor)`  
Get latex trace options from Data

`export_to_tikz_groupGraphs(theGraphs: optimeed.core.graphs.Graphs, foldername, additionalPreamble=lambda: "", additionalAxisOptions=lambda graphId: "", additionalTraceOptions=lambda graphId, traceId: "", debug=False)`  
Export the graphs as group

#### Parameters

- **theGraphs** – Graphs to save
- **foldername** – Foldername to save
- **additionalPreamble** – method that returns string for custom tikz options
- **additionalAxisOptions** – method that returns string for custom tikz options
- **additionalTraceOptions** – method that returns string for custom tikz options

#### Returns

`do_preamble3D()`

`format_Griddata(X, Y, Z)`

`format_scatterdata(x, y, z)`

`export_to_tikz_contour_plot(list_of_traces3, foldername, filename_data='data')`  
Export the graphs as group

#### Parameters

- **list\_of\_traces3** – List of 3D traces
- **foldername** – Foldername to save

- **filename\_data** – filename of the data

#### Returns

tools

## Module Contents

### Classes

### Functions

### Attributes

`_workspace_path`

`class text_format`

```
PURPLE = [95m
CYAN = [96m
DARKCYAN = [36m
BLUE = [94m
GREEN = [92m
YELLOW = [93m
WHITE = [30m
RED = [91m
BOLD = [1m
UNDERLINE = [4m
END = [0m
```

`software_version()`

`find_and_replace(begin_char, end_char, theStr, replace_function)`

`create_unique_dirname(dirname)`

Create dirname if it doesn't exists, otherwise append an integer to dirname and create it.

**Parameters** **dirname** – name of the directory to create

**Returns** name of the directory created

`applyEquation(objectIn, s)`

Apply literal expression based on an object

#### Parameters

- **objectIn** – Object
- **s** – literal expression. Float variables taken from the object are written between {}, int between []. Example: s="{x}+{y}\*2" if x and y are attributes of objectIn.

**Returns** value (float)

**arithmeticEval** (*s*)

**isNonePrintMessage** (*theObject, theMessage, show\_type=SHOW\_INFO*)

**getPath\_workspace** ()

Get workspace path (i.e., location where optimeed files will be created). Create directory if doesn't exist.

**setPath\_workspace** (*thePath*)

Set workspace path (i.e., location where optimeed files will be created)

**getLineInfo** (*lvl=1*)

**printIfShown** (*theStr, show\_type=SHOW\_DEBUG, isToPrint=True, appendTypeName=True, end='n'*)

**universalPath** (*thePath*)

**add\_suffix\_to\_path** (*thePath, suffix*)

**get\_object\_attrs** (*obj*)

**rsetattr** (*obj, attr, val*)

setattr, but recursively. Works with list (i.e. theObj.myList[0].var\_x)

**rgetattr** (*obj, attr*)

getattr, but recursively. Works with list.

**indentParagraph** (*text\_in, indent\_level=1*)

Add ' ' at beginning of strings and after each ' '.

**truncate** (*theStr, truncsize*)

**get\_recursive\_attrs** (*theObject, max\_recursion\_level=2*)

**str\_all\_attr** (*theObject, max\_recursion\_level*)

**get\_2D\_pareto** (*xList, yList, max\_X=True, max\_Y=True*)

Get 2D pareto front

#### Parameters

- **xList** – list of x coordinates
- **yList** – list of y coordinates
- **max\_X** – True if x is to maximize
- **max\_Y** – true if y is to maximize

**Returns** x pareto-optimal coordinates, y pareto-optimal coordinates, indices of these points in input parameters

**get\_ND\_pareto** (*objectives\_list, are\_maxobjectives\_list=None*)

Return the N-D pareto front

#### Parameters

- **objectives\_list** – list of list of objectives: example `[[0,1], [1,1], [2,2]]`
- **are\_maxobjectives\_list** – for each objective, tells if they are to be maximized or not: example `[True, False]`. Default: False

**Returns** extracted\_pareto, indices: list of `[x, y, ...]` points forming the pareto front, and list of the indices of these points from the base list.

**delete\_indices\_from\_list** (*indices, theList*)

Delete elements from list at indices

#### Parameters

- **indices** – list

- **theList** – list

**merge\_two\_dicts** (*dict1, dict2*)

Merge two dicts without affecting them

**Returns** new dictionary

**deep\_sizeof** (*obj*)

**order\_lists** (*ref\_list, linked\_list*)

## Package Contents

## Classes

## Functions

## Attributes

**\_workspace\_path**

**class text\_format**

**PURPLE** = [95m

**CYAN** = [96m

**DARKCYAN** = [36m

**BLUE** = [94m

**GREEN** = [92m

**YELLOW** = [93m

**WHITE** = [30m

**RED** = [91m

**BOLD** = [1m

**UNDERLINE** = [4m

**END** = [0m

**software\_version** ()

**find\_and\_replace** (*begin\_char, end\_char, theStr, replace\_function*)

**create\_unique\_dirname** (*dirname*)

Create dirname if it doesn't exists, otherwise append an integer to dirname and create it.

**Parameters** **dirname** – name of the directory to create

**Returns** name of the directory created

**applyEquation** (*objectIn, s*)

Apply literal expression based on an object

**Parameters**

- **objectIn** – Object
- **s** – literal expression. Float variables taken from the object are written between {}, int between []. Example: `s="{x}+{y}*2"` if `x` and `y` are attributes of `objectIn`.

**Returns** value (float)

**arithmeticEval** (*s*)

**isNonePrintMessage** (*theObject*, *theMessage*, *show\_type=SHOW\_INFO*)

**getPath\_workspace** ()

Get workspace path (i.e., location where optimeed files will be created). Create directory if doesn't exist.

**setPath\_workspace** (*thePath*)

Set workspace path (i.e., location where optimeed files will be created)

**getLineInfo** (*lvl=1*)

**printIfShown** (*theStr*, *show\_type=SHOW\_DEBUG*, *isToPrint=True*, *appendTypeName=True*, *end='n'*)

**universalPath** (*thePath*)

**add\_suffix\_to\_path** (*thePath*, *suffix*)

**get\_object\_attrs** (*obj*)

**rsetattr** (*obj*, *attr*, *val*)

setattr, but recursively. Works with list (i.e. `theObj.myList[0].var_x`)

**rgetattr** (*obj*, *attr*)

getattr, but recursively. Works with list.

**indentParagraph** (*text\_in*, *indent\_level=1*)

Add ' ' at beginning of strings and after each ' '.

**truncate** (*theStr*, *truncsize*)

**get\_recursive\_attrs** (*theObject*, *max\_recursion\_level=2*)

**str\_all\_attr** (*theObject*, *max\_recursion\_level*)

**get\_2D\_pareto** (*xList*, *yList*, *max\_X=True*, *max\_Y=True*)

Get 2D pareto front

#### Parameters

- **xList** – list of x coordinates
- **yList** – list of y coordinates
- **max\_X** – True if x is to maximize
- **max\_Y** – true if y is to maximize

**Returns** x pareto-optimal coordinates, y pareto-optimal coordinates, indices of these points in input parameters

**get\_ND\_pareto** (*objectives\_list*, *are\_maxobjectives\_list=None*)

Return the N-D pareto front

#### Parameters

- **objectives\_list** – list of list of objectives: example `[[0,1], [1,1], [2,2]]`
- **are\_maxobjectives\_list** – for each objective, tells if they are to be maximized or not: example `[True, False]`. Default: False

**Returns** `extracted_pareto`, `indices`: list of `[x, y, ...]` points forming the pareto front, and list of the indices of these points from the base list.

**`delete_indices_from_list`** (*indices*, *theList*)

Delete elements from list at indices

**Parameters**

- **`indices`** – list
- **`theList`** – list

**`merge_two_dicts`** (*dict1*, *dict2*)

Merge two dicts without affecting them

**Returns** new dictionary

**`deep_sizeof`** (*obj*)

**`order_lists`** (*ref\_list*, *linked\_list*)

**`SHOW_WARNING`** = 0

**`SHOW_INFO`** = 1

**`SHOW_ERROR`** = 2

**`SHOW_DEBUG`** = 3

**`SHOW_LOGS`** = 4

**`SHOW_CURRENT`**

**`setCurrentShow`** (*show\_types*)

Change text type to be displayed by `PrintfShown`

**`getCurrentShow`** ()

Get text type to be displayed by `PrintfShown`

**`disableLogs`** ()

Disable all logs

**`enableLogs`** ()

Show all logs

**`SHOW_WARNING`** = 0

**`SHOW_INFO`** = 1

**`SHOW_ERROR`** = 2

**`SHOW_DEBUG`** = 3

**`SHOW_LOGS`** = 4

**`SHOW_CURRENT`**

**`setCurrentShow`** (*show\_types*)

Change text type to be displayed by `PrintfShown`

**`getCurrentShow`** ()

Get text type to be displayed by `PrintfShown`

**`disableLogs`** ()

Disable all logs

**`enableLogs`** ()

Show all logs

**getPath\_workspace()**

Get workspace path (i.e., location where optimeed files will be created). Create directory if doesn't exist.

**obj\_to\_json** (*theObj*)

Extract the json dictionary from the object. The data saved are automatically detected, using typehints. ex: x: int=5 will be saved, x=5 won't. Inheritance of annotation is managed by this function

**json\_to\_obj** (*json\_dict*)

Convenience class to create object from dictionary. Only works if CLASS\_TAG is valid

**Parameters** *json\_dict* – dictionary loaded from a json file.

**Raises**

- **TypeError** – if class can not be found
- **KeyError** – if CLASS\_TAG not present in dictionary

**json\_to\_obj\_safe** (*json\_dict, cls*)

Safe class to create object from dictionary.

**Parameters**

- *json\_dict* – dictionary loaded from a json file
- *cls* – class object to instantiate with dictionary

**encode\_str\_json** (*theStr*)

**decode\_str\_json** (*theStr*)

**get\_json\_module\_tree\_from\_dict** (*jsonDict*)

Return dict containing {CLASS\_TAG: "class\_name", MODULE\_TAG: "module\_name", "attribute1": {"class\_name": "module\_name", ...}}

**remove\_module\_tree\_from\_string** (*theStr*)

Used to compress string by removing `__module__` and `__class__` entries (used with `get_json_module_tree_from_dict`)

**apply\_module\_tree\_to\_dict** (*nestedTree, nestedObject, raiseError=False*)

Restore `__module__` and `__class__` entries from nestedTree in nestedDict

**indentParagraph** (*text\_in, indent\_level=1*)

Add ' ' at beginning of strings and after each ' '.

**rgetattr** (*obj, attr*)

getattr, but recursively. Works with list.

**printIfShown** (*theStr, show\_type=SHOW\_DEBUG, isToPrint=True, appendTypeName=True, end='n')*

**SHOW\_WARNING** = 0

**SHOW\_DEBUG** = 3

**SHOW\_INFO** = 1

**SHOW\_ERROR** = 2

**delete\_indices\_from\_list** (*indices, theList*)

Delete elements from list at indices

**Parameters**

- *indices* – list
- *theList* – list



```

class SingleObjectSaveLoad
class DataStruct_Interface

    __str__()
        Return str(self).

class ListDataStruct_Interface
    Bases: DataStruct_Interface

    get_list_attributes (attributeName)
        Get the value of attributeName of all the data in the Collection

        Parameters attributeName – string (name of the attribute to get)

        Returns list

class AutosaveStruct (dataStruct, filename="", change_filename_if_exists=True)
    Structure that provides automated save of DataStructures

    __str__()
        Return str(self).

    get_filename ()
        Get set filename

    set_filename (filename, change_filename_if_exists)

        Parameters

        • filename – Filename to set

        • change_filename_if_exists – If already exists, create a new filename

    stop_autosave ()
        Stop autosave

    start_autosave (timer_autosave, safe_save=True)
        Start autosave

    save (safe_save=True)
        Save

    get_datastruct ()
        Return :class:`~DataStruct_Interface`

    __getstate__ ()

    __setstate__ (state)

class ListDataStruct (compress_save=False)
    Bases: ListDataStruct_Interface

    _DATA_STR = data

    _COMPRESS_SAVE_STR = module_tree

    __len__ ()

    get_length ()

    clone (filename)
        Clone the datastructure to a new location

    save (filename)
        Save data using json format. The data to be saved are automatically detected, see obj_to_json()

```

**extract\_collection\_from\_indices** (*indices*)  
Extract data from the collection at specific indices, and return it as new collection

**\_format\_str\_save** ()  
Save data using json format. The data to be saved are automatically detected, see *obj\_to\_json* ()

**\_format\_data\_lines** ()

**\_get\_json\_module\_tree** ()

**add\_data** (*data\_in*)  
Add a data to the list

**get\_data** ()  
Get full list of datas

**get\_data\_generator** ()  
Get a generator to all the data stored

**get\_data\_at\_index** (*index*)

**set\_data** (*theData*)  
Set full list of datas

**set\_data\_at\_index** (*data\_in*, *index*)  
Replace data at specific index

**reset\_data** ()

**delete\_points\_at\_indices** (*indices*)  
Delete several elements from the Collection

**Parameters** *indices* – list of indices to delete

**merge** (*collection*)  
Merge a collection with the current collection

**Parameters** *collection* – Collection to merge

**get\_nbr\_elements** ()

**Returns** the number of elements contained inside the structure

**theLock**

**class Performance\_ListDataStruct** (*stack\_size=500*)  
Bases: *ListDataStruct\_Interface*

**\_NBR\_ELEMENTS** = *nbr\_elements*

**\_STACK\_SIZE** = *stack\_size*

**\_COMPRESS\_SAVE\_STR** = *module\_tree*

**\_initialize** (*filename*)

**\_get\_list\_from\_file** (*filenumber*)

**extract\_collection\_from\_indices** (*indices*)  
Extract data from the collection at specific indices, and return it as new collection

**clone** (*filename*)  
Clone the datastructure to a new location

**\_get\_str\_mainfile** ()

**get\_total\_nbr\_elements** (*count\_unsaved=True*)

**add\_data** (*theData*)

Add data to the collection

**add\_json\_data** (*theStr*)

Add already deserialized data to the collection

**\_save\_moduletree** (*theDict*)

**\_map\_index\_to\_file** (*index*)

**\_get\_json\_str\_at\_index** (*index*, *refresh\_cache=False*)

Internal method to return the json string at index

**reorder** (*permutations*)

Reorder collection accordingly to permutations. E.G, list\_of\_indices = [0,3,2] with collection elems [0,2,1]  
=> collection elems = [0,2,3] :param permutations: :return: /

**get\_data\_at\_index** (*index*, *ignore\_attributes=None*, *none\_if\_error=False*)

Same as parent, with additional kwargs

#### Parameters

- **index** –
- **ignore\_attributes** – ignore attributes to deserialize (list)
- **none\_if\_error** –

#### Returns

**save** (*filename*)

Save the datastructure to filename

**get\_data\_generator** (*\*\*kwargs*)

**get\_nbr\_elements** ()

**Returns** the number of elements contained inside the structure

**set\_data\_at\_index** (*data\_in*, *index*)

Replace data at specific index

**set\_data\_at\_indices** (*data\_list*, *indices*)

Replace datas at specific indices :param data\_list: list of objects to set to the collection, at specific indices  
:param indices: list of indices :return:

**delete\_points\_at\_indices** (*indices*)

Delete several elements from the Collection

**Parameters** **indices** – list of indices to delete

**default\_palette** (*N*)

**blackOnly** (*N*)

**dark2** (*N*)

**printIfShown** (*theStr*, *show\_type=SHOW\_DEBUG*, *isToPrint=True*, *appendTypeName=True*, *end='n'*)

**SHOW\_WARNING** = 0

**convert\_color\_with\_alpha** (*color*, *alpha=255*)

Same as meth:*convert\_color* but with transparency

```
class Data (x: list, y: list, x_label="", y_label="", legend="", is_scattered=False, transfo_x=lambda self-
    Data, x: x, transfo_y=lambda selfData, y: y, xlim=None, ylim=None, permutations=None,
    sort_output=False, color=None, alpha=255, symbol='o', symbolsize=8, fillsymbol=True, out-
    linesymbol=1.8, linestyle='-', width=2, meta=None)
```

This class is used to store informations necessary to plot a 2D graph. It has to be combined with a gui to be useful (ex. pyqtgraph)

**set\_kwargs** (kwargs)

Set a kwarg after creation of the class

**set\_data** (x: list, y: list)

Overwrites current datapoints with new set

**set\_meta** (meta)

Set associated 'Z' data

**get\_x** ()

Get x coordinates of datapoints

**get\_symbolsize** ()

Get size of the symbols

**symbol\_isfilled** ()

Check if symbols has to be filled or not

**get\_symbolOutline** ()

Get color factor of outline of symbols

**get\_length\_data** ()

Get number of points

**get\_xlim** ()

Get x limits of viewbox

**get\_ylim** ()

Get y limits of viewbox

**get\_y** ()

Get y coordinates of datapoints

**get\_meta** ()

Get associated 'Z' data

**get\_color** ()

Get color of the line, without transformation

**get\_color\_alpha** ()

Get color of the line. Return r, g, b in 0, 255 scale

**get\_alpha** ()

Get opacity

**get\_width** ()

Get width of the line

**get\_number\_of\_points** ()

Get number of points

**get\_plot\_data** ()

Call this method to get the x and y coordinates of the points that have to be displayed. => After transformation, and after permutations.

**Returns** x (list), y (list)

**get\_plot\_meta** (*x, y*)

Call this method to get the z coordinates of the points that been displayed. => After transformation, and after permutations.

**Returns** z (list)

**get\_permutations** (*x=None*)

Return the transformation 'permutation': `xplot[i] = xdata[permutation[i]]`

**get\_invert\_permutations** ()

Return the inverse of permutations: `xdata[i] = xplot[revert[i]]`

**get\_dataIndex\_from\_graphIndex** (*index\_graph\_point*)

From an index given in graph, recovers the index of the data.

**Parameters** *index\_graph\_point* – Index in the graph

**Returns** index of the data

**get\_dataIndices\_from\_graphIndices** (*index\_graph\_point\_list*)

Same as `get_dataIndex_from_graphIndex` but with a list in entry. Can (?) improve performances for huge dataset.

**Parameters** *index\_graph\_point\_list* – List of Index in the graph

**Returns** List of index of the data

**get\_graphIndex\_from\_dataIndex** (*index\_data*)

From an index given in the data, recovers the index of the graph.

**Parameters** *index\_data* – Index in the data

**Returns** index of the graph

**get\_graphIndices\_from\_dataIndices** (*index\_data\_list*)

Same as `get_graphIndex_from_dataIndex` but with a list in entry. Can (?) improve performances for huge dataset.

**Parameters** *index\_data\_list* – List of Index in the data

**Returns** List of index of the graph

**set\_permutations** (*permutations*)

Set permutations between datapoints of the trace

**Parameters** *permutations* – list of indices to plot (example: [0, 2, 1] means that the first point will be plotted, then the third, then the second one)

**get\_x\_label** ()

Get x label of the trace

**get\_y\_label** ()

Get y label of the trace

**get\_legend** ()

Get name of the trace

**get\_symbol** ()

Get symbol

**add\_point** (*x, y*)

Add point(s) to trace (inputs can be list or numeral)

**delete\_point** (*index\_point*)

Delete a point from the datapoints

**isScattered()**  
Check if plot is scatteded

**set\_indices\_points\_to\_plot** (*indices*)  
Set indices points to plot

**get\_indices\_points\_to\_plot** ()  
Get indices points to plot

**get\_linestyle** ()  
Get linestyle

**\_\_str\_\_** ()  
Return str(self).

**export\_str** ()  
Method to save the points constituting the trace

**set\_color** (*theColor*)  
Set trace color

**set\_legend** (*theLegend*)  
Set legend

**class Graph**

Simple graph container that contains several traces

**add\_trace** (*data*)  
Add a trace to the graph

**Parameters** *data* – *Data*

**Returns** id of the created trace

**remove\_trace** (*idTrace*)  
Delete a trace from the graph

**Parameters** *idTrace* – id of the trace to delete

**get\_trace** (*idTrace*) → *Data*  
Get data object of *idTrace*

**Parameters** *idTrace* – id of the trace to get

**Returns** *Data*

**get\_all\_traces** ()  
Get all the traces id of the graph

**get\_all\_traces\_ids** ()  
Get all the traces id of the graph :return: list of id graphs

**export\_str** ()

**class Graphs**

Contains several *Graph*

**updateChildren** ()

**add\_trace\_firstGraph** (*data*, *updateChildren=True*)  
Same as **add\_trace**, but only if graphs has only one id :param *data*: :param *updateChildren*: :return:

**add\_trace** (*idGraph*, *data*, *updateChildren=True*)  
Add a trace to the graph

**Parameters**

- **idGraph** – id of the graph
- **data** – *Data*
- **updateChildren** – Automatically calls callback functions

**Returns** id of the created trace

**remove\_trace** (*idGraph*, *idTrace*, *updateChildren=True*)

Remove the trace from the graph

**Parameters**

- **idGraph** – id of the graph
- **idTrace** – id of the trace to remove
- **updateChildren** – Automatically calls callback functions

**get\_first\_graph** ()

Get id of the first graph

**Returns** id of the first graph

**get\_graph** (*idGraph*)

Get graph object at idgraph

**Parameters** **idGraph** – id of the graph to get

**Returns** *Graph*

**get\_all\_graphs\_ids** ()

Get all ids of the graphs

**Returns** list of id graphs

**get\_all\_graphs** ()

Get all graphs. Return dict {id: *Graph*}

**add\_graph** (*updateChildren=True*)

Add a new graph

**Returns** id of the created graph

**remove\_graph** (*idGraph*)

Delete a graph

**Parameters** **idGraph** – id of the graph to delete

**add\_update\_method** (*childObject*)

Add a callback each time a graph is modified.

**Parameters** **childObject** – method without arguments

**export\_str** ()

Export all the graphs in text

**Returns** str

**merge** (*otherGraphs*)

**reset** ()

**is\_empty** ()

**griddata\_found** = True

```
class Plot3D_Generic(x_label="", y_label="", z_label="", legend="", x_lim=None, y_lim=None,
                    z_lim=None)

    get_lim(axis)
    get_label(axis)
    get_legend()

class GridPlot_Generic(X, Y, Z, **kwargs)
    Bases: Plot3D_Generic

    get_plot_data()

class ContourPlot(*args, **kwargs)
    Bases: GridPlot_Generic

    get_levels()
    get_number_of_contours()

class FilledContourPlot(*args, **kwargs)
    Bases: ContourPlot

class SurfPlot(X, Y, Z, **kwargs)
    Bases: GridPlot_Generic

class MeshPlot(X, Y, Z, **kwargs)
    Bases: GridPlot_Generic

class ScatterPlot3(x, y, z, **kwargs)
    Bases: Plot3D_Generic

    get_plot_data()
    get_color()

convert_to_gridplot(x, y, z, x_interval=None, y_interval=None, n_x=20, n_y=20)
    Convert set of points x, y, z to a grid
```

#### Parameters

- **x** –
- **y** –
- **z** –
- **x\_interval** – [Min, max] of the grid. If none, use min and max values
- **y\_interval** – [Min, max] of the grid. If none, use min and max values
- **n\_x** – number of points in x direction
- **n\_y** – number of points in y direction

Returns X, Y, Z as grid

```
class HowToPlotGraph(attribute_x, attribute_y, kwargs_graph=None, check_if_plot_elem=None,
                    meta=None)

    __str__()
        Return str(self).

class LinkDataGraph
```



**add\_collection** (*theCollection*, *kwargs=None*)

Add a collection (that will be a future trace)

**Parameters**

- **theCollection** –
- **kwargs** – kwargs associated with the collection (e.g., color, symbol style, etc.)

**Returns** unique id associated with the collection

**remove\_collection** (*collectionId*)

Remove collection from the graphs

**Parameters** **collectionId** – ID of the collection

**Returns**

**set\_shadow\_collection** (*master\_collectionId*, *shadow\_collection*)

Link a collection to an other

**Parameters**

- **master\_collectionId** – ID of the collection that is displayed in the graph
- **shadow\_collection** – collection to link to the master.

**Returns**

**get\_graphs** ()

**get\_howToPlotGraph** (*idGraph*)

**add\_graph** (*howToPlotGraph*)

Add new graph to be plotted.

**Parameters** **howToPlotGraph** – *HowToPlotGraph*

**Returns**

**get\_idCollections** ()

Get all ids of the plotted collections

**get\_idGraphs** ()

Get all ids of the graphs

**get\_idTraces** (*idGraph*)

Get all ids of the traces of graph \$idGraph

**get\_idCollection\_from\_graph** (*idGraph*, *idTrace*)

Get id of collection plotted in graph \$idGraph and trace \$idTrace

**get\_collection** (*idCollection*, *getShadow=True*)

**update\_graphs** ()

Update the graphs: update graphs, traces, and X-Y data

**get\_collection\_from\_graph** (*idGraph*, *idTrace*, *getShadow=True*) → opti-  
meed.core.ListDataStruct\_Interface

From indices in the graph, get corresponding collection

**get\_clicked\_item** (*idGraph*, *idTrace*, *idPoint*, *getShadow=True*)

Get the data hidden behind the clicked point

**Parameters**

- **idGraph** – ID of the graph

- **idTrace** – ID of the trace
- **idPoint** – ID of the point
- **getShadow** – If true, will return the data from the collection linked to the collection that is plotted

**Returns** Object in collection

**get\_clicked\_items** (*idGraph*, *idTrace*, *idPoint\_list*, *getShadow=True*)

Same as **get\_clicked\_item**, but using a list of points

**delete\_clicked\_item** (*idGraph*, *idTrace*, *idPoint*)

Remove item from the collection

**delete\_clicked\_items** (*idGraph*, *idTrace*, *idPoints*)

Same, but for a list of points

**get\_graph\_and\_trace\_from\_idCollection** (*idCollection*)

Reverse search: from a collection, get all associated graphs

**get\_idcollection\_from\_collection** (*theCollection*)

Reverse search: from a collection, find its id

**get\_idPoints\_from\_indices\_in\_collection** (*idGraph*, *idTrace*, *indices\_in\_collection*)

From indices in a collection, find the associated idPoints of the graph

**class Base\_Option** (*name*, *based\_value*, *choices=None*)

**get\_value** ()

**get\_name** ()

**set\_value** (*value*)

**get\_choices** ()

**class Option\_bool** (*name*, *based\_value*, *choices=None*)

Bases: *Base\_Option*

**name** :str

**value** :bool

**set\_value** (*value*)

**get\_choices** ()

**class Option\_str** (*name*, *based\_value*, *choices=None*)

Bases: *Base\_Option*

**name** :str

**value** :str

**set\_value** (*value*)

**class Option\_int** (*name*, *based\_value*, *choices=None*)

Bases: *Base\_Option*

**name** :str

**value** :int

**set\_value** (*value*)

```

class Option_float (name, based_value, choices=None)
    Bases: Base_Option

    name :str
    value :float
    set_value (value)

class Option_dict (name, based_value, choices=None)
    Bases: Base_Option

    name :str
    value :dict
    set_value (value)

class Option_class

    options_bool :Dict[int, Option_bool]
    options_str :Dict[int, Option_str]
    options_int :Dict[int, Option_int]
    options_float :Dict[int, Option_float]
    options_dict :Dict[int, Option_dict]
    add_option (idOption, theOption)
    get_option_name (idOption)
    get_option_value (idOption)
    set_option (idOption, value)
    _pack_options ()
    __str__ ()
        Return str(self).

has_scipy = True

class fast_LUT_interpolation (independent_variables, dependent_variables)
    Class designed for fast interpolation in look-up table when successive searches are called often. Otherwise use
    griddata

    interpolate (point, fill_value=np.nan)
        Perform the interpolation :param point: coordinates to interpolate (tuple or list of tuples for multipoints)
        :param fill_value: value to put if extrapolated. :return: coordinates

    interpolate_table (x0, x_values, y_values)
        From sorted table (x,y) find y0 corresponding to x0 (linear interpolation)

    derivate (t, y)

    linspace (start, stop, npoints)

    reconstitute_signal (amplitudes, phases, numberOfPeriods=1, x_points=None, n_points=50)
        Reconstitute the signal from fft. Number of periods of the signal must be specified if different of 1

    my_fft (y)
        Real FFT of signal Bx, with real amplitude of harmonics. Input signal must be within a period.

    cart2pol (x, y)

```

**pol2cart** (*rho, phi*)

**partition** (*array, begin, end*)

**quicksort** (*array*)

**dist** (*p, q*)

Return the Euclidean distance between points p and q. :param p: [x, y] :param q: [x, y] :return: distance (float)

**sparse\_subset** (*points, r*)

Returns a maximal list of elements of points such that no pairs of points in the result have distance less than r.  
:param points: list of tuples (x,y) :param r: distance :return: corresponding subset (list), indices of the subset (list)

**integrate** (*x, y*)

Performs Integral(x[0] to x[-1]) of y dx

**Parameters**

- **x** – x axis coordinates (list)
- **y** – y axis coordinates (list)

**Returns** integral value

**my\_fourier** (*x, y, n, L*)

Fourier analys

**Parameters**

- **x** – x axis coordinates
- **y** – y axis coordinates
- **n** – number of considered harmonic
- **L** – half-period length

**Returns** a and b coefficients ( $y = a*\cos(x) + b*\sin(y)$ )

**get\_ellipse\_axes** (*a, b, dphi*)

Trouve les longueurs des axes majeurs et mineurs de l'ellipse, ainsi que l'orientation de l'ellipse. ellipse:  $x(t) = A*\cos(t)$ ,  $y(t) = B*\cos(t+dphi)$  Etapes: longueur demi ellipse CENTRÉE =  $\sqrt{a^2 \cos^2(x) + b^2 \cos^2(t+phi)}$   
Minimisation de cette formule => obtention formule  $\tan(2x) = \alpha/\beta$

**convert\_color** (*color*)

Convert a color to a tuple if color is a char, otherwise return the tuple.

**Parameters** **color** – (r,g,b) or char.

**Returns**

**convert\_color\_with\_alpha** (*color, alpha=255*)

Same as meth:*convert\_color* but with transparency

**rgetattr** (*obj, attr*)

getattr, but recursively. Works with list.

**rsetattr** (*obj, attr, val*)

setattr, but recursively. Works with list (i.e. theObj.myList[0].var\_x)

**printIfShown** (*theStr, show\_type=SHOW\_DEBUG, isToPrint=True, appendTypeName=True, end='n'*)

**SHOW\_ERROR** = 2

**SHOW\_WARNING** = 0

**MODULE\_TAG** = `__module__`

**CLASS\_TAG** = `__class__`

**EXCLUDED\_TAGS**

**getExecPath()**

**class SaveableObject**

Abstract class for dynamically type-hinted objects. This class is to solve the special case where the exact type of an attribute is not known before runtime, yet has to be saved.

**get\_additional\_attributes\_to\_save()**

Return list of attributes corresponding to object, whose type cannot be determined statically (e.g. topology change)

**get\_additional\_attributes\_to\_save\_list()**

Same behavior as `get_additional_attributes_to_save`, but where the attributes contains list of unknown items

**\_\_isclass**(*theObject*)

Extends the default `isclass` method with typing

**get\_type\_class**(*typ*)

Get the type of the class. used to compare objects from Typing.

**\_\_get\_object\_class**(*theObj*)

**\_\_get\_object\_module**(*theObj*)

**\_\_object\_to\_FQCN**(*theobj*)

Gets module path of object

**\_\_find\_class**(*moduleName, className*)

**json\_to\_obj**(*json\_dict*)

Convenience class to create object from dictionary. Only works if **CLASS\_TAG** is valid

**Parameters** **json\_dict** – dictionary loaded from a json file.

**Raises**

- **TypeError** – if class can not be found
- **KeyError** – if **CLASS\_TAG** not present in dictionary

**json\_to\_obj\_safe**(*json\_dict, cls*)

Safe class to create object from dictionary.

**Parameters**

- **json\_dict** – dictionary loaded from a json file
- **cls** – class object to instantiate with dictionary

**\_\_instantiates\_annotated\_object**(*\_json\_dict, \_cls*)

**\_\_get\_annotations**(*theObj*)

Return annotated attributes (*theObj* being the type of the object)

**obj\_to\_json**(*theObj*)

Extract the json dictionary from the object. The data saved are automatically detected, using typehints. ex: `x: int=5` will be saved, `x=5` won't. Inheritance of annotation is managed by this function

**\_\_get\_attributes\_to\_save**(*theObj*)

Return list (attribute, `is_first`)

**get\_json\_module\_tree\_from\_dict** (*jsonDict*)

Return dict containing {CLASS\_TAG: "class\_name", MODULE\_TAG: "module\_name", "attribute1": {"class\_name": "module\_name", ...}}

**remove\_module\_tree\_from\_string** (*theStr*)

Used to compress string by removing `__module__` and `__class__` entries (used with `get_json_module_tree_from_dict`)

**apply\_module\_tree\_to\_dict** (*nestedTree, nestedObject, raiseError=False*)

Restore `__module__` and `__class__` entries from `nestedTree` in `nestedDict`

**encode\_str\_json** (*theStr*)

**decode\_str\_json** (*theStr*)

**export\_to\_tikz\_groupGraphs** (*theGraphs: optimeed.core.graphs.Graphs, foldername, additionalPreamble=*`lambda: "`*additionalAxisOptions=*`lambda graphId: "`*additionalTraceOptions=*`lambda graphId, traceId: "`*debug=False*`)`

Export the graphs as group

#### Parameters

- **theGraphs** – Graphs to save
- **foldername** – Foldername to save
- **additionalPreamble** – method that returns string for custom tikz options
- **additionalAxisOptions** – method that returns string for custom tikz options
- **additionalTraceOptions** – method that returns string for custom tikz options

#### Returns

**export\_to\_tikz\_contour\_plot** (*list\_of\_traces3, foldername, filename\_data='data'*)

Export the graphs as group

#### Parameters

- **list\_of\_traces3** – List of 3D traces
- **foldername** – Foldername to save
- **filename\_data** – filename of the data

#### Returns

**printIfShown** (*theStr, show\_type=SHOW\_DEBUG, isToPrint=True, appendTypeName=True, end='n'*)

**SHOW\_WARNING** = 0

**get\_path\_to\_inkscape** ()

**get\_inkscape\_version** ()

**inkscape\_version**

**inkscape\_svg\_to\_pdf** (*filename\_svg, filename\_pdf*)

**inkscape\_svg\_to\_png** (*filename\_svg, filename\_png*)

## optimize

## Subpackages

## characterization

### characterization

#### Module Contents

#### Classes

##### **class** Characterization

Bases: `optimeed.optimize.characterization.interfaceCharacterization.InterfaceCharacterization`

Interface for the evaluation of a device

**compute** (*theDevice*)

Action to perform to characterize (= compute the objective function) of the device.

**Parameters** **theDevice** – the device to characterize

##### **interfaceCharacterization**

#### Module Contents

#### Classes

##### **class** InterfaceCharacterization

Interface for the evaluation of a device

**\_\_str\_\_** ()

Return str(self).

#### Package Contents

#### Classes

##### **class** InterfaceCharacterization

Interface for the evaluation of a device

**\_\_str\_\_** ()

Return str(self).

##### **class** Characterization

Bases: `optimeed.optimize.characterization.interfaceCharacterization.InterfaceCharacterization`

Interface for the evaluation of a device

**compute** (*theDevice*)

Action to perform to characterize (= compute the objective function) of the device.

**Parameters** **theDevice** – the device to characterize

## mathsToPhysics

`interfaceMathsToPhysics`

### Module Contents

#### Classes

**class** `InterfaceMathsToPhysics`

Interface to transform output from the optimizer to meaningful variables of the device

`mathsToPhysics`

### Module Contents

#### Classes

**class** `MathsToPhysics`

Bases: `optimeed.optimize.mathsToPhysics.interfaceMathsToPhysics.InterfaceMathsToPhysics`

Dummy yet powerful example of maths to physics. The optimization variables are directly injected to the device

**fromMathsToPhys** (*xVector*, *theDevice*, *theOptimizationVariables*)

Transforms an input vector coming from the optimization (e.g. [0.23, 4, False]) to “meaningful” variable (ex: length, number of poles, flag).

#### Parameters

- **xVector** – List of optimization variables from the optimizer
- **theDevice** – `InterfaceDevice`
- **opti\_variables** – list of `OptimizationVariable`

**fromPhysToMaths** (*theDevice*, *theOptimizationVariables*)

Extracts a mathematical vector from meaningful variable of the Device

#### Parameters

- **theDevice** – `InterfaceDevice`
- **opti\_variables** – list of `OptimizationVariable`

**Returns** List of optimization variables

`__str__` ()

Return str(self).

### Package Contents



## Classes

### **class MathsToPhysics**

Bases: `optimeed.optimize.mathsToPhysics.interfaceMathsToPhysics.InterfaceMathsToPhysics`

Dummy yet powerful example of maths to physics. The optimization variables are directly injected to the device

**fromMathsToPhys** (*xVector, theDevice, theOptimizationVariables*)

Transforms an input vector coming from the optimization (e.g. [0.23, 4, False]) to “meaningful” variable (ex: length, number of poles, flag).

#### Parameters

- **xVector** – List of optimization variables from the optimizer
- **theDevice** – InterfaceDevice
- **opti\_variables** – list of OptimizationVariable

**fromPhysToMaths** (*theDevice, theOptimizationVariables*)

Extracts a mathematical vector from meaningful variable of the Device

#### Parameters

- **theDevice** – InterfaceDevice
- **opti\_variables** – list of OptimizationVariable

**Returns** List of optimization variables

**\_\_str\_\_** ()

Return str(self).

### **class InterfaceMathsToPhysics**

Interface to transform output from the optimizer to meaningful variables of the device

## objAndCons

### fastObjCons

## Module Contents

## Classes

### **class FastObjCons** (*constraintEquation, name=None*)

Bases: `optimeed.optimize.objAndCons.interfaceObjCons.InterfaceObjCons`

Convenience class to create an objective or a constraint very fast.

**compute** (*theDevice*)

Get the value of the objective or the constraint. The objective is to MINIMIZE and the constraint has to respect VALUE <= 0

**Parameters** **theDevice** – Input device that has already been evaluated

**Returns** float.

**get\_name** ()

interfaceObjCons

## Module Contents

### Classes

**class** InterfaceObjCons

Interface class for objectives and constraints. The objective is to MINIMIZE and the constraint has to respect  $VALUE \leq 0$

**get\_name** ()

**\_\_str\_\_** ()

Return str(self).

## Package Contents

### Classes

**class** FastObjCons (*constraintEquation*, *name=None*)

Bases: *optimeed.optimize.objAndCons.interfaceObjCons.InterfaceObjCons*

Convenience class to create an objective or a constraint very fast.

**compute** (*theDevice*)

Get the value of the objective or the constraint. The objective is to MINIMIZE and the constraint has to respect  $VALUE \leq 0$

**Parameters** **theDevice** – Input device that has already been evaluated

**Returns** float.

**get\_name** ()

**class** InterfaceObjCons

Interface class for objectives and constraints. The objective is to MINIMIZE and the constraint has to respect  $VALUE \leq 0$

**get\_name** ()

**\_\_str\_\_** ()

Return str(self).

## optiAlgorithms

### Subpackages

#### convergence

#### evolutionaryConvergence

## Module Contents

## Classes

### class EvolutionaryConvergence

Bases: `optimeed.optimize.optiAlgorithms.convergence.interfaceConvergence.InterfaceConvergence`

convergence class for population-based algorithm

**objectives\_per\_step** :Dict[int, List[List[float]]]

**constraints\_per\_step** :Dict[int, List[List[float]]]

**paretos\_per\_step** :Dict[int, List[List[float]]]

**hypervolume\_per\_step** :Dict[int, List[float]]

**set\_curr\_step** (*theObjectives\_list*, *theConstraints\_list*)

**\_extract\_N\_steps** (*N*)

**get\_pareto\_convergence** (*max\_number\_of\_points=None*)

**get\_pareto\_at\_step** (*step*)

**get\_hypervolume** (*pareto*, *refPoint=None*)

**get\_hypervolume\_convergence** (*max\_number\_of\_points*)

**get\_nadir\_point** (*pareto*)

**last\_step** ()

**get\_nb\_objectives** ()

**get\_scalar\_convergence\_evolution** (*max\_number\_of\_points*)

**get\_graphs** (*max\_number\_of\_points=None*)

Return *Graphs*

### hypervolume

## Module Contents

### Classes

### Attributes

**\_\_author\_\_** = Simon Wessing

### class HyperVolume (*referencePoint*)

Hypervolume computation based on variant 3 of the algorithm in the paper: C. M. Fonseca, L. Paquete, and M. Lopez-Ibanez. An improved dimension-sweep algorithm for the hypervolume indicator. In IEEE Congress on Evolutionary Computation, pages 1157-1163, Vancouver, Canada, July 2006.

Minimization is implicitly assumed here!

**compute** (*front*)

Returns the hypervolume that is dominated by a non-dominated front.

Before the HV computation, front and reference point are translated, so that the reference point is [0, ..., 0].

**hvRecursive** (*dimIndex, length, bounds*)

Recursive call to hypervolume calculation.

In contrast to the paper, the code assumes that the reference point is  $[0, \dots, 0]$ . This allows the avoidance of a few operations.

**preProcess** (*front*)

Sets up the list data structure needed for calculation.

**sortByDimension** (*nodes, i*)

Sorts the list of nodes by the *i*-th value of the contained points.

**class MultiList** (*numberLists*)

A special data structure needed by FonsecaHyperVolume.

It consists of several doubly linked lists that share common nodes. So, every node has multiple predecessors and successors, one in every list.

**class Node** (*numberLists, cargo=None*)

**\_\_str\_\_** ()

Return str(self).

**\_\_str\_\_** ()

Return str(self).

**\_\_len\_\_** ()

Returns the number of lists that are included in this MultiList.

**getLength** (*i*)

Returns the length of the *i*-th list.

**append** (*node, index*)

Appends a node to the end of the list at the given index.

**extend** (*nodes, index*)

Extends the list at the given index with the nodes.

**remove** (*node, index, bounds*)

Removes and returns 'node' from all lists in  $[0, \text{'index'}[$ .

**reinsert** (*node, index, bounds*)

Inserts 'node' at the position it had in all lists in  $[0, \text{'index'}[$  before it was removed. This method assumes that the next and previous nodes of the node that is reinserted are in the list.

**interfaceConvergence**

## Module Contents

### Classes

**class InterfaceConvergence**

Simple interface to visually get the convergence of any optimization problem

### Package Contents

## Classes

### class EvolutionaryConvergence

Bases: `optimeed.optimize.optiAlgorithms.convergence.interfaceConvergence.InterfaceConvergence`

convergence class for population-based algorithm

**objectives\_per\_step** :Dict[int, List[List[float]]]

**constraints\_per\_step** :Dict[int, List[List[float]]]

**paretos\_per\_step** :Dict[int, List[List[float]]]

**hypervolume\_per\_step** :Dict[int, List[float]]

**set\_curr\_step** (*theObjectives\_list*, *theConstraints\_list*)

**\_extract\_N\_steps** (*N*)

**get\_pareto\_convergence** (*max\_number\_of\_points=None*)

**get\_pareto\_at\_step** (*step*)

**get\_hypervolume** (*pareto*, *refPoint=None*)

**get\_hypervolume\_convergence** (*max\_number\_of\_points*)

**get\_nadir\_point** (*pareto*)

**last\_step** ()

**get\_nb\_objectives** ()

**get\_scalar\_convergence\_evolution** (*max\_number\_of\_points*)

**get\_graphs** (*max\_number\_of\_points=None*)

Return *Graphs*

### class InterfaceConvergence

Simple interface to visually get the convergence of any optimization problem

## pyswarm

## pso

## Module Contents

### Classes

### Functions

**\_is\_feasible** (*theList*)

**\_format\_fx\_fs** (*objectives\_pop*, *constraints\_pop*)

**class MyMapEvaluator** (*evaluation\_function*, *callback\_on\_evaluation*)

**evaluate\_all** (*x*)

```
class MyMultiprocessEvaluator (evaluation_function, callback_on_evaluation, numberOfCores)
```

```
    evaluate_all (x)
```

```
pso (lb, ub, initialVectorGuess, theEvaluator, maxtime, callback_generation=lambda objectives, constraints:
    None, swarmsize=100, omega=0.5, phip=0.5, phig=0.5)
    Perform a particle swarm optimization (PSO)
```

**lb: list** Lower bounds of each parameter

**ub: list** upper bounds of each parameter

**initialVectorGuess: list** initial vector guess for the solution (to be included inside population)

theEvaluator : object define before maxtime : float

The maximum time (in s) before stopping the algorithm

**callback\_generation: function lambda (bjectives (as list), constraints (as list)) per step** Useful to log convergence

**swarmsize** [int] The number of particles in the swarm (Default: 100)

**omega** [scalar] Particle velocity scaling factor (Default: 0.5)

**phip** [scalar] Scaling factor to search away from the particle's best known position (Default: 0.5)

**phig** [scalar] Scaling factor to search away from the swarm's best known position (Default: 0.5)

**g** [array] The swarm's best known position (optimal design)

**f** [scalar] The objective value at g

## Package Contents

### Classes

### Functions

```
_is_feasible (theList)
```

```
_format_fx_fs (objectives_pop, constraints_pop)
```

```
class MyMapEvaluator (evaluation_function, callback_on_evaluation)
```

```
    evaluate_all (x)
```

```
class MyMultiprocessEvaluator (evaluation_function, callback_on_evaluation, numberOfCores)
```

```
    evaluate_all (x)
```

```
pso (lb, ub, initialVectorGuess, theEvaluator, maxtime, callback_generation=lambda objectives, constraints:
    None, swarmsize=100, omega=0.5, phip=0.5, phig=0.5)
    Perform a particle swarm optimization (PSO)
```

**lb: list** Lower bounds of each parameter

**ub: list** upper bounds of each parameter

**initialVectorGuess: list** initial vector guess for the solution (to be included inside population)

**theEvaluator** : object define before maxtime : float

The maximum time (in s) before stopping the algorithm

**callback\_generation: function lambda (bjectives (as list), constraints (as list)) per step** Useful to log convergence

**swarmsize** [int] The number of particles in the swarm (Default: 100)

**omega** [scalar] Particle velocity scaling factor (Default: 0.5)

**phip** [scalar] Scaling factor to search away from the particle's best known position (Default: 0.5)

**phig** [scalar] Scaling factor to search away from the swarm's best known position (Default: 0.5)

**g** [array] The swarm's best known position (optimal design)

**f** [scalar] The objective value at  $g$

## NLOpt\_Algorithm

### Module Contents

#### Classes

**class ConvergenceManager**

**add\_point** (*newObj*)

**set\_pop\_size** (*popSize*)

**class NLOpt\_Algorithm**

Bases: *optimeed.optimize.optiAlgorithms.algorithmInterface.AlgorithmInterface, optimeed.core.Option\_class*

Interface for the optimization algorithm

**ALGORITHM** = 0

**POPULATION\_SIZE** = 1

**initialize** (*initialVectorGuess, listOfOptimizationVariables*)

This function is called once parameters can't be changed anymore, before "get\_convergence".

#### Parameters

- **initialVectorGuess** – list of variables that describe the initial individual
- **listOfOptimizationVariables** – list of *optimeed.optimize.optiVariable.OptimizationVariable*

#### Returns

**compute** ()

Launch the optimization

**Returns** vector of optimal variables

**set\_evaluationFunction** (*evaluationFunction, callback\_on\_evaluate, numberOfObjectives, \_numberOfConstraints*)

Set the evaluation function and all the necessary callbacks

**Parameters**

- **evaluationFunction** – check `evaluateObjectiveAndConstraints()`
- **callback\_on\_evaluation** – check `callback_on_evaluation()`. Call this function after performing the evaluation of the individuals
- **numberOfObjectives** – int, number of objectives
- **numberOfConstraints** – int, number of constraints

**set\_maxtime** (*maxTime*)

Set maximum optimization time (in seconds)

**\_\_str\_\_** ()

Return str(self).

**get\_convergence** ()

Get the convergence of the optimization

**Returns** *InterfaceConvergence***algorithmInterface****Module Contents****Classes****class AlgorithmInterface**

Interface for the optimization algorithm

**reset** ()**monobjective\_PSO****Module Contents****Classes****class Monobjective\_PSO**Bases: *optimeed.optimize.optiAlgorithms.algorithmInterface.  
AlgorithmInterface, optimeed.core.Option\_class*

Interface for the optimization algorithm

**NUMBER\_OF\_CORES** = 1**initialize** (*initialVectorGuess, listOfOptimizationVariables*)

This function is called once parameters can't be changed anymore, before "get\_convergence".

**Parameters**

- **initialVectorGuess** – list of variables that describe the initial individual
- **listOfOptimizationVariables** – list of *optimeed.optimize.  
optiVariable.OptimizationVariable*

**Returns**



**compute()**

Launch the optimization

**Returns** vector of optimal variables

**set\_evaluationFunction** (*evaluationFunction*, *callback\_on\_evaluate*, *numberOfObjectives*, *\_numberOfConstraints*)

Set the evaluation function and all the necessary callbacks

**Parameters**

- **evaluationFunction** – check `evaluateObjectiveAndConstraints()`
- **callback\_on\_evaluation** – check `callback_on_evaluation()`. Call this function after performing the evaluation of the individuals
- **numberOfObjectives** – int, number of objectives
- **numberOfConstraints** – int, number of constraints

**set\_maxtime** (*maxTime*)

Set maximum optimization time (in seconds)

**\_\_str\_\_()**

Return `str(self)`.

**get\_convergence()**

Get the convergence of the optimization

**Returns** *InterfaceConvergence*

## multiObjective\_GA

### Module Contents

#### Classes

**class MyProblem** (*theOptimizationVariables*, *nbr\_objectives*, *nbr\_constraints*, *evaluationFunction*)

Bases: `optimeed.optimize.optiAlgorithms.platypus.core.Problem`

Automatically sets the optimization problem

**evaluate** (*solution*)

Evaluates the problem.

By default, this method calls the function passed to the constructor. Alternatively, a problem can subclass and override this method. When overriding, this method is responsible for updating the objectives and constraints stored in the solution.

**solution:** **Solution** The solution to evaluate.

**class MyGenerator** (*initialVectorGuess*)

Bases: `optimeed.optimize.optiAlgorithms.platypus.Generator`

Population generator to insert initial individual

**generate** (*problem*)

**class MaxTimeTerminationCondition** (*maxTime*)

Bases: `optimeed.optimize.optiAlgorithms.platypus.core.TerminationCondition`

Abstract class for defining termination conditions.

**initialize** (*algorithm*)

Initializes this termination condition.

This method is used to collect any initial state, such as the current NFE or current time, needed for calculating the termination criteria.

**algorithm** [Algorithm] The algorithm being run.

**shouldTerminate** (*algorithm*)

Checks if the algorithm should terminate.

Check the termination condition, returning True if the termination condition is satisfied; False otherwise. This method is called after each iteration of the algorithm.

**algorithm** [Algorithm] The algorithm being run.

**class ConvergenceTerminationCondition** (*minrelchange\_percent=0.1, nb\_generation=15*)

Bases: `optimeed.optimize.optiAlgorithms.platypus.core.TerminationCondition`

Abstract class for defining termination conditions.

**initialize** (*algorithm*)

Initializes this termination condition.

This method is used to collect any initial state, such as the current NFE or current time, needed for calculating the termination criteria.

**algorithm** [Algorithm] The algorithm being run.

**shouldTerminate** (*algorithm*)

Checks if the algorithm should terminate.

Check the termination condition, returning True if the termination condition is satisfied; False otherwise. This method is called after each iteration of the algorithm.

**algorithm** [Algorithm] The algorithm being run.

**class SeveralTerminationCondition**

Bases: `optimeed.optimize.optiAlgorithms.platypus.core.TerminationCondition`

Abstract class for defining termination conditions.

**initialize** (*algorithm*)

Initializes this termination condition.

This method is used to collect any initial state, such as the current NFE or current time, needed for calculating the termination criteria.

**algorithm** [Algorithm] The algorithm being run.

**add** (*theTerminationCondition*)

**shouldTerminate** (*algorithm*)

Checks if the algorithm should terminate.

Check the termination condition, returning True if the termination condition is satisfied; False otherwise. This method is called after each iteration of the algorithm.

**algorithm** [Algorithm] The algorithm being run.

**class MyMapEvaluator** (*callback\_on\_evaluation*)

Bases: `optimeed.optimize.optiAlgorithms.platypus.evaluator.Evaluator`

**evaluate\_all** (*jobs, \*\*kwargs*)

```

class MyMultiprocessEvaluator (callback_on_evaluation, numberOfCores)
    Bases: optimeed.optimize.optiAlgorithms.platypus.evaluator.Evaluator

    my_callback (output)

    evaluate_all (jobs, **kwargs)

    close ()

class MultiObjective_GA
    Bases: optimeed.optimize.optiAlgorithms.algorithmInterface.
           AlgorithmInterface, optimeed.core.Option_class

    Based on Platypus Library. Workflow: Define what to optimize and which function to call with a Problem
    Define the initial population with a Generator Define the algorithm. As options, define how to evaluate
    the elements with a Evaluator, i.e., for multiprocessing. Define what is the termination condition of the
    algorithm with TerminationCondition. Here, termination condition is a maximum time.

    DIVISION_OUTER = 0

    OPTI_ALGORITHM = 1

    NUMBER_OF_CORES = 2

    KWARGS_ALGO = 3

    initialize (initialVectorGuess, listOfOptimizationVariables)
        This function is called just before running optimization algorithm.

    compute ()
        Launch the optimization

        Returns vector of optimal variables

    set_evaluationFunction (evaluationFunction, callback_on_evaluation, numberOfObjectives,
                           numberOfConstraints)
        Set the evaluation function and all the necessary callbacks

    Parameters

        • evaluationFunction – check evaluateObjectiveAndConstraints ()

        • callback_on_evaluation – check callback_on_evaluation (). Call this
          function after performing the evaluation of the individuals

        • numberOfObjectives – int, number of objectives

        • numberOfConstraints – int, number of constraints

    set_maxtime (maxTime)
        Set maximum optimization time (in seconds)

    __str__ ()
        Return str(self).

    get_convergence ()
        This function is called just before compute. Because the convergence is contained in opti algorithm, it
        must be created now.

    add_terminationCondition (theTerminationCondition)

    reset ()

```

## Package Contents

### Classes

#### **class MultiObjective\_GA**

Bases: `optimeed.optimize.optiAlgorithms.algorithmInterface.AlgorithmInterface`, `optimeed.core.Option_class`

Based on [Platypus Library](#). Workflow: Define what to optimize and which function to call with a `Problem`. Define the initial population with a `Generator`. Define the algorithm. As options, define how to evaluate the elements with a `Evaluator`, i.e., for multiprocessing. Define what is the termination condition of the algorithm with `TerminationCondition`. Here, termination condition is a maximum time.

**DIVISION\_OUTER** = 0

**OPTI\_ALGORITHM** = 1

**NUMBER\_OF\_CORES** = 2

**KWARGS\_ALGO** = 3

**initialize** (*initialVectorGuess*, *listOfOptimizationVariables*)

This function is called just before running optimization algorithm.

**compute** ()

Launch the optimization

**Returns** vector of optimal variables

**set\_evaluationFunction** (*evaluationFunction*, *callback\_on\_evaluation*, *numberOfObjectives*, *numberOfConstraints*)

Set the evaluation function and all the necessary callbacks

#### **Parameters**

- **evaluationFunction** – check `evaluateObjectiveAndConstraints()`
- **callback\_on\_evaluation** – check `callback_on_evaluation()`. Call this function after performing the evaluation of the individuals
- **numberOfObjectives** – int, number of objectives
- **numberOfConstraints** – int, number of constraints

**set\_maxtime** (*maxTime*)

Set maximum optimization time (in seconds)

**\_\_str\_\_** ()

Return str(self).

**get\_convergence** ()

This function is called just before compute. Because the convergence is contained in opti algorithm, it must be created now.

**add\_terminationCondition** (*theTerminationCondition*)

**reset** ()

#### **class Monobjective\_PSO**

Bases: `optimeed.optimize.optiAlgorithms.algorithmInterface.AlgorithmInterface`, `optimeed.core.Option_class`

Interface for the optimization algorithm

**NUMBER\_OF\_CORES = 1**

**initialize** (*initialVectorGuess*, *listOfOptimizationVariables*)

This function is called once parameters can't be changed anymore, before "get\_convergence".

#### Parameters

- **initialVectorGuess** – list of variables that describe the initial individual
- **listOfOptimizationVariables** – list of `optimeed.optimize.OptiVariable.OptimizationVariable`

#### Returns

**compute** ()

Launch the optimization

**Returns** vector of optimal variables

**set\_evaluationFunction** (*evaluationFunction*, *callback\_on\_evaluate*, *numberOfObjectives*, *\_numberOfConstraints*)

Set the evaluation function and all the necessary callbacks

#### Parameters

- **evaluationFunction** – check `evaluateObjectiveAndConstraints()`
- **callback\_on\_evaluation** – check `callback_on_evaluation()`. Call this function after performing the evaluation of the individuals
- **numberOfObjectives** – int, number of objectives
- **numberOfConstraints** – int, number of constraints

**set\_maxtime** (*maxTime*)

Set maximum optimization time (in seconds)

**\_\_str\_\_** ()

Return str(self).

**get\_convergence** ()

Get the convergence of the optimization

**Returns** `InterfaceConvergence`

## optiHistoric

### Module Contents

#### Classes

**class OptiHistoric** (*optiname='opti'*, *autosave\_timer=60 \* 5*, *autosave=True*, *create\_new\_directory=True*, *performance\_datastruct=True*)

Contains all the points that have been evaluated

**class \_pointData** (*currTime*, *objectives*, *constraints*)

**time** :float

**objectives** :List[float]

**constraints** :List[float]

```
class _LogParams
```

```
    add_parameters (params)
```

```
    get_rows_indices (list_of_params)
```

```
log_after_evaluation (returned_values: dict)
```

Save the output of evaluate to optiHistoric. This function should be called by the optimizer IN a process safe context.

```
set_results (devicesList)
```

```
get_best_devices_without_reevaluating (list_of_best_params)
```

```
set_convergence (theConvergence)
```

```
save ()
```

```
get_convergence ()
```

Returns convergence *InterfaceConvergence*

```
get_devices ()
```

Returns List of devices (ordered by evaluation number)

```
get_logopti ()
```

Returns Log optimization (to check the convergence)

```
start (optimization_parameters)
```

Function called upon starting the optimization. Create folders.

**optiVariable**

## Module Contents

### Classes

```
class OptimizationVariable (attributeName)
```

Contains information about the optimization of a variable

```
    attributeName :str
```

```
    get_attribute_name ()
```

Return the attribute to set

```
    add_prefix_attribute_name (thePrefix)
```

Used for nested object, lower the name by prefix. Example: R\_ext becomes (thePrefix).R\_ext

```
    get_PhysToMaths (deviceIn)
```

Convert the initial value of the variable contained in the device to optimization variable value

Parameters **deviceIn** – *InterfaceDevice*

Returns value of the corresponding optimization variable

```
    do_MathsToPhys (variableValue, deviceIn)
```

Apply the value to the device

```
    __str__ ()
```

Return str(self).

```

class Real_OptimizationVariable (attributeName, val_min, val_max)
    Bases: OptimizationVariable

    Real (continuous) optimization variable. Most used type

    val_min :float
    val_max :float
    get_min_value ()
    get_max_value ()
    get_PhysToMaths (deviceIn)
        Convert the initial value of the variable contained in the device to optimization variable value

        Parameters deviceIn – InterfaceDevice

        Returns value of the corresponding optimization variable

    do_MathsToPhys (value, deviceIn)
        Apply the value to the device

    __str__ ()
        Return str(self).

class Binary_OptimizationVariable (attributeName)
    Bases: OptimizationVariable

    Boolean (True/False) optimization variable.

    get_PhysToMaths (deviceIn)
        Convert the initial value of the variable contained in the device to optimization variable value

        Parameters deviceIn – InterfaceDevice

        Returns value of the corresponding optimization variable

    do_MathsToPhys (value, deviceIn)
        Apply the value to the device

    __str__ ()
        Return str(self).

class Integer_OptimizationVariable (attributeName, val_min, val_max)
    Bases: OptimizationVariable

    Integer variable, in [min_value, max_value]

    val_min :int
    val_max :int
    get_min_value ()
    get_max_value ()
    get_PhysToMaths (deviceIn)
        Convert the initial value of the variable contained in the device to optimization variable value

        Parameters deviceIn – InterfaceDevice

        Returns value of the corresponding optimization variable

    do_MathsToPhys (value, deviceIn)
        Apply the value to the device

```

```
__str__ ()  
    Return str(self).
```

**optimizer**

## Module Contents

### Classes

### Functions

### Attributes

#### default

```
class OptimizerSettings (theDevice, theObjectives, theConstraints, theOptimizationVariables,  
                        theOptimizationAlgorithm=None, theMathsToPhysics=None, theCharac-  
                        terization=None)
```

Bases: *optimeed.core.SaveableObject*

Abstract class for dynamically type-hinted objects. This class is to solve the special case where the exact type of an attribute is not known before runtime, yet has to be saved.

```
get_additional_attributes_to_save ()  
    Return list of attributes corresponding to object, whose type cannot be determined statically (e.g. topology  
    change)
```

```
get_additional_attributes_to_save_list ()  
    Same behavior as get_additional_attributes_to_save, but where the attributes contains list of unknown  
    items
```

```
get_device ()
```

```
get_M2P ()
```

```
get_charac ()
```

```
get_optivariabiles ()
```

```
get_objectives ()
```

```
get_constraints ()
```

```
get_optialgorithm ()
```

```
class _Evaluator (optimization_parameters: OptimizerSettings)
```

This is the main class that serves as evaluator. This class is NOT process safe (i.e., copy of it might be generated upon process call)

```
start ()
```

```
evaluate (x)  
    Evaluates the performances of device associated to entrance vector x. Outputs the objective function and  
    the constraints, and other data used in optiHistoric.
```

This function is NOT process safe: “self.” is a FORK in multiprocessing algorithms. It means that the motor originally contained in self. is modified only in the fork, and only gathered by reaching the end of the fork.

**Parameters** **x** – Input mathematical vector from optimization algorithm



**Returns** dictionary, containing objective values (list of scalar), constraint values (list of scalar), and other info (motor, time)

**reevaluate\_solutions** (*x\_solutions*)

**run\_optimization** (*optimization\_parameters: OptimizerSettings, opti\_historic, max\_opti\_time\_sec=10*)  
Perform the optimization.

**Returns** list of the best optimized devices, convergence information

## Package Contents

### Classes

### Functions

**class InterfaceCharacterization**

Interface for the evaluation of a device

**\_\_str\_\_** ()

Return str(self).

**class Characterization**

Bases: *optimeed.optimize.characterization.interfaceCharacterization.InterfaceCharacterization*

Interface for the evaluation of a device

**compute** (*theDevice*)

Action to perform to characterize (= compute the objective function) of the device.

**Parameters** **theDevice** – the device to characterize

**class MathsToPhysics**

Bases: *optimeed.optimize.mathsToPhysics.interfaceMathsToPhysics.InterfaceMathsToPhysics*

Dummy yet powerful example of maths to physics. The optimization variables are directly injected to the device

**fromMathsToPhys** (*xVector, theDevice, theOptimizationVariables*)

Transforms an input vector coming from the optimization (e.g. [0.23, 4, False]) to “meaningful” variable (ex: length, number of poles, flag).

**Parameters**

- **xVector** – List of optimization variables from the optimizer
- **theDevice** – InterfaceDevice
- **opti\_variables** – list of OptimizationVariable

**fromPhysToMaths** (*theDevice, theOptimizationVariables*)

Extracts a mathematical vector from meaningful variable of the Device

**Parameters**

- **theDevice** – InterfaceDevice
- **opti\_variables** – list of OptimizationVariable

**Returns** List of optimization variables

```
__str__()
    Return str(self).
```

**class InterfaceMathsToPhysics**

Interface to transform output from the optimizer to meaningful variables of the device

**class FastObjCons** (*constraintEquation, name=None*)

Bases: *optimeed.optimize.objAndCons.interfaceObjCons.InterfaceObjCons*

Convenience class to create an objective or a constraint very fast.

**compute** (*theDevice*)

Get the value of the objective or the constraint. The objective is to MINIMIZE and the constraint has to respect VALUE <= 0

**Parameters** *theDevice* – Input device that has already been evaluated

**Returns** float.

**get\_name** ()

**class InterfaceObjCons**

Interface class for objectives and constraints. The objective is to MINIMIZE and the constraint has to respect VALUE <= 0

**get\_name** ()

```
__str__()
    Return str(self).
```

**class MultiObjective\_GA**

Bases: *optimeed.optimize.optiAlgorithms.algorithmInterface.AlgorithmInterface, optimeed.core.Option\_class*

Based on [Platypus Library](#). Workflow: Define what to optimize and which function to call with a Problem Define the initial population with a Generator Define the algorithm. As options, define how to evaluate the elements with a Evaluator, i.e., for multiprocessing. Define what is the termination condition of the algorithm with TerminationCondition. Here, termination condition is a maximum time.

**DIVISION\_OUTER** = 0

**OPTI\_ALGORITHM** = 1

**NUMBER\_OF\_CORES** = 2

**KWARGS\_ALGO** = 3

**initialize** (*initialVectorGuess, listOfOptimizationVariables*)

This function is called just before running optimization algorithm.

**compute** ()

Launch the optimization

**Returns** vector of optimal variables

**set\_evaluationFunction** (*evaluationFunction, callback\_on\_evaluation, numberOfObjectives, numberOfConstraints*)

Set the evaluation function and all the necessary callbacks

**Parameters**

- **evaluationFunction** – check `evaluateObjectiveAndConstraints()`
- **callback\_on\_evaluation** – check `callback_on_evaluation()`. Call this function after performing the evaluation of the individuals

- **numberOfObjectives** – int, number of objectives
- **numberOfConstraints** – int, number of constraints

**set\_maxtime** (*maxTime*)

Set maximum optimization time (in seconds)

**\_\_str\_\_** ()

Return str(self).

**get\_convergence** ()

This function is called just before compute. Because the convergence is contained in opti algorithm, it must be created now.

**add\_terminationCondition** (*theTerminationCondition*)

**reset** ()

**class Monobjective\_PSO**

Bases: `optimeed.optimize.optiAlgorithms.algorithmInterface.AlgorithmInterface`, `optimeed.core.Option_class`

Interface for the optimization algorithm

**NUMBER\_OF\_CORES** = 1

**initialize** (*initialVectorGuess*, *listOfOptimizationVariables*)

This function is called once parameters can't be changed anymore, before "get\_convergence".

#### Parameters

- **initialVectorGuess** – list of variables that describe the initial individual
- **listOfOptimizationVariables** – list of `optimeed.optimize.optiVariable.OptimizationVariable`

#### Returns

**compute** ()

Launch the optimization

**Returns** vector of optimal variables

**set\_evaluationFunction** (*evaluationFunction*, *callback\_on\_evaluate*, *numberOfObjectives*, *\_numberOfConstraints*)

Set the evaluation function and all the necessary callbacks

#### Parameters

- **evaluationFunction** – check `evaluateObjectiveAndConstraints` ()
- **callback\_on\_evaluation** – check `callback_on_evaluation` (). Call this function after performing the evaluation of the individuals
- **numberOfObjectives** – int, number of objectives
- **numberOfConstraints** – int, number of constraints

**set\_maxtime** (*maxTime*)

Set maximum optimization time (in seconds)

**\_\_str\_\_** ()

Return str(self).

**get\_convergence** ()

Get the convergence of the optimization

Returns *InterfaceConvergence*

**class Real\_OptimizationVariable** (*attributeName, val\_min, val\_max*)

Bases: OptimizationVariable

Real (continuous) optimization variable. Most used type

**val\_min** :float

**val\_max** :float

**get\_min\_value** ()

**get\_max\_value** ()

**get\_PhysToMaths** (*deviceIn*)

Convert the initial value of the variable contained in the device to optimization variable value

**Parameters** **deviceIn** – InterfaceDevice

**Returns** value of the corresponding optimization variable

**do\_MathsToPhys** (*value, deviceIn*)

Apply the value to the device

**\_\_str\_\_** ()

Return str(self).

**class Binary\_OptimizationVariable** (*attributeName*)

Bases: OptimizationVariable

Boolean (True/False) optimization variable.

**get\_PhysToMaths** (*deviceIn*)

Convert the initial value of the variable contained in the device to optimization variable value

**Parameters** **deviceIn** – InterfaceDevice

**Returns** value of the corresponding optimization variable

**do\_MathsToPhys** (*value, deviceIn*)

Apply the value to the device

**\_\_str\_\_** ()

Return str(self).

**class Integer\_OptimizationVariable** (*attributeName, val\_min, val\_max*)

Bases: OptimizationVariable

Integer variable, in [min\_value, max\_value]

**val\_min** :int

**val\_max** :int

**get\_min\_value** ()

**get\_max\_value** ()

**get\_PhysToMaths** (*deviceIn*)

Convert the initial value of the variable contained in the device to optimization variable value

**Parameters** **deviceIn** – InterfaceDevice

**Returns** value of the corresponding optimization variable

**do\_MathsToPhys** (*value, deviceIn*)

Apply the value to the device

```

__str__()
    Return str(self).

run_optimization (optimization_parameters: OptimizerSettings, opti_historic, max_opti_time_sec=10)
    Perform the optimization.

    Returns list of the best optimized devices, convergence information

class OptimizerSettings (theDevice, theObjectives, theConstraints, theOptimizationVariables,
                        theOptimizationAlgorithm=None, theMathsToPhysics=None, theCharac-
                        terization=None)
    Bases: optimeed.core.SaveableObject

    Abstract class for dynamically type-hinted objects. This class is to solve the special case where the exact type
    of an attribute is not known before runtime, yet has to be saved.

    get_additional_attributes_to_save ()
        Return list of attributes corresponding to object, whose type cannot be determined statically (e.g. topology
        change)

    get_additional_attributes_to_save_list ()
        Same behavior as get_additional_attributes_to_save, but where the attributes contains list of unknown
        items

    get_device ()

    get_M2P ()

    get_charac ()

    get_optivariabls ()

    get_objectives ()

    get_constraints ()

    get_optialgorithm ()

class OptiHistoric (optiname='opti', autosave_timer=60 * 5, autosave=True, cre-
                    ate_new_directory=True, performance_datastruct=True)
    Contains all the points that have been evaluated

class _pointData (currTime, objectives, constraints)

    time :float
    objectives :List[float]
    constraints :List[float]

class _LogParams

    add_parameters (params)

    get_rows_indices (list_of_params)

log_after_evaluation (returned_values: dict)
    Save the output of evaluate to optiHistoric. This function should be called by the optimizer IN a process
    safe context.

set_results (devicesList)

get_best_devices_without_reevaluating (list_of_best_params)

set_convergence (theConvergence)

```

**save()**

**get\_convergence()**

**Returns** convergence *InterfaceConvergence*

**get\_devices()**

**Returns** List of devices (ordered by evaluation number)

**get\_logopti()**

**Returns** Log optimization (to check the convergence)

**start(optimization\_parameters)**

Function called upon starting the optimization. Create folders.

**visualize**

**Subpackages**

**graphs**

**colormap\_pyqtgraph**

**Module Contents**

**Functions**

**Attributes**

**has\_matplotlib = True**

**sequence**

**matplotlib\_colormap\_to\_pg\_colormap(colormap\_name, n\_ticks=16)**

**cmapToColormap(cmap, nTicks=16)**

Converts a Matplotlib cmap to pyqtgraphs colormaps. No dependency on matplotlib. Parameters:

*cmap*: Cmap object. Imported from matplotlib.cm.\* *nTicks*: Number of ticks to create when dict of functions is used. Otherwise unused.

author: Sebastian Hofer

**graphVisual**

**Module Contents**

**Classes**

**class GraphVisual(theWidgetGraphVisual)**

Provide an interface to a graph. A graph contains traces.

**set\_fontTicks(fontSize, fontname=None)**

Set font of the ticks

**Parameters**

- **fontSize** – Size of the font
- **fontname** – Name of the font

**set\_numberTicks** (*number, axis*)

Set the number of ticks to be displayed

**Parameters**

- **number** – Number of ticks for the axis
- **axis** – Axis (string, “bottom”, “left”, “right”, “top”)

**Returns**

**set\_fontLabel** (*fontSize, color='#000', fontname=None*)

Set font of the axis labels

**Parameters**

- **fontSize** – font size
- **color** – color in hexadecimal (str)
- **fontname** – name of the font

**get\_legend** () → optimeed.visualize.graphs.pyqtgraphRedefine.myLegend

Get the legend

**get\_axis** (*axis*) → optimeed.visualize.graphs.pyqtgraphRedefine.myAxis

Get the axis

**Parameters** **axis** – Axis (string, “bottom”, “left”, “right”, “top”)

**Returns** axis object

**set\_fontLegend** (*font\_size, font\_color, fontname=None*)

**set\_label\_pos** (*orientation, x\_offset=0, y\_offset=0*)

**set\_color\_palette** (*palette*)

**apply\_palette** ()

**hide\_axes** ()

**add\_feature** (*theFeature*)

To add any pyqtgraph item to the graph

**remove\_feature** (*theFeature*)

To remove any pyqtgraph item from the graph

**add\_data** (*idGraph, theData*)

**set\_graph\_properties** (*theTrace*)

This function is automatically called on creation of the graph

**set\_lims** (*xlim, ylim*)

Set limits of the graphs, xlim or ylim = [val\_low, val\_high]. Or None.

**add\_trace** (*idTrace, theTrace*)

Add a TraceVisual to the graph, with index idTrace

**set\_legend** ()

Set default legend options (color and font)

**set\_title** (*titleName*, *\*\*kwargs*)  
Set title of the graph

**Parameters** **titleName** – title to set

**get\_trace** (*idTrace*) → optimeed.visualize.graphs.traceVisual.TraceVisual  
Return the TraceVisual correspondong to the index idTrace

**get\_all\_traces** ()  
Return a dictionary {idtrace: TraceVisual}.

**delete\_trace** (*idTrace*)  
Delete the trace of index idTrace

**delete** ()  
Delete the graph

**linkXToGraph** (*graph*)  
Link the axis of the current graph to an other *GraphVisual*

**update** ()  
Update the traces contained in the graph

**fast\_update** ()  
Same as *update* () but faster. This is NOT thread safe (cannot be called a second time before finishing operation)

**axis\_equal** ()

**log\_mode** (*x=False*, *y=False*)

**grid\_off** ()  
Turn off grid

## pyqtgraphRedefine

### Module Contents

#### Classes

#### Attributes

#### isOnWindows

Other modified files (directly): ScatterPlotItem.py, to change point selection. Ctrl + clic: select area. Clic: only one single point:

#### class OnClicSelector:

```
def __init__(self): self.p_list = []

def add_point(self, newp): self.p_list.append(newp)

def draw(self, painter):
    if len(self.p_list) > 2: pen = fn.mkPen(1) pen.setWidthF(2) painter.setPen(pen)
    painter.drawPolyline(QtGui.QPolygonF(self.p_list))

def reset(self): self.p_list = []

def getPath(self): return path.Path([(p.x(), p.y()) for p in self.p_list] + [(self.p_list[-1].x(), self.p_list[-1].y())])
```



```
def mouseDragEvent(self, ev):
```

```
    if ev.modifiers() and QtCore.Qt.ControlModifier: ev.accept()
```

```
        self.clicSelector.add_point(ev.pos()) if ev.isFinish():
```

```
            path = self.clicSelector.getPath() points = self.points() contains_points =
            path.contains_points([(p.pos().x(), p.pos().y()) for p in points]) indices = [i for i,
            cond in enumerate(contains_points) if cond] points_clicked = [points[i] for i in
            indices] self.ptsClicked = points_clicked self.sigClicked.emit(self, self.ptsClicked)
            self.clicSelector.reset()
```

```
        self.update()
```

```
    else: ev.ignore()
```

```
class myGraphicsLayoutWidget (parent=None, **_kwargs)
```

```
    Bases: optimeed.visualize.graphs.pyqtgraph.GraphicsView
```

Re-implementation of QGraphicsView that removes scrollbars and allows unambiguous control of the viewed coordinate range. Also automatically creates a GraphicsScene and a central QGraphicsWidget that is automatically scaled to the full view geometry.

This widget is the basis for PlotWidget, GraphicsLayoutWidget, and the view widget in ImageView.

By default, the view coordinate system matches the widget's pixel coordinates and automatically updates when the view is resized. This can be overridden by setting `autoPixelRange=False`. The exact visible range can be set with `setRange()`.

The view can be panned using the middle mouse button and scaled using the right mouse button if enabled via `enableMouse()` (but ordinarily, we use `ViewBox` for this functionality).

```
useOpenGL (b=True)
```

```
    Overwrited to fix bad antialiasing while using openGL
```

```
class myGraphicsLayout
```

```
    Bases: optimeed.visualize.graphs.pyqtgraph.GraphicsLayout
```

Used for laying out GraphicsWidgets in a grid. This is usually created automatically as part of a GraphicsWindow or GraphicsLayoutWidget.

```
addItem (item, row=None, col=None, rowspan=1, colspan=1)
```

```
    Add an item to the layout and place it in the next available cell (or in the cell specified). The item must be
    an instance of a QGraphicsWidget subclass.
```

```
set_graph_disposition (item, row=1, col=1, rowspan=1, colspan=1)
```

```
    Function to modify the position of an item in the list
```

#### Parameters

- **item** – WidgetPlotItem to set
- **row** – Row
- **col** – Column
- **rowspan** –
- **colspan** –

#### Returns

```
class myItemSample (item)
```

```
    Bases: optimeed.visualize.graphs.pyqtgraph.graphicsItems.LegendItem.
    ItemSample
```

Class responsible for drawing a single item in a LegendItem (sans label)

**set\_offset** (*offset*)

**set\_width\_cell** (*width*)

**paint** (*p, \*args*)

Overwrites to make matlab-like samples

**class myLegend** (*size=None, offset=(30, 30), is\_light=False*)

Bases: `optimeed.visualize.graphs.pyqtgraph.LegendItem`

Legend that fixes bugs (flush left + space) from pyqtgraph's legend

**set\_space\_sample\_label** (*theSpace*)

To set the gap between the sample and the label

**set\_offset\_sample** (*offset*)

To tune the offset between the sample and the text

**set\_width\_cell\_sample** (*width*)

Set width of sample

**updateSize** ()

**addItem** (*item, name*)

Overwrites to flush left

**apply\_width\_sample** ()

**set\_font** (*font\_size, font\_color, fontname=None*)

**paint** (*p, \*args*)

Overwrited to select background color

**set\_position** (*position, offset*)

Set the position of the legend, in a corner.

#### Parameters

- **position** – String (NW, NE, SW, SE), indicates which corner the legend is close
- **offset** – Tuple (xoff, yoff), x and y offset from the edge

#### Returns

**class myLabelItem** (*text= ' ', parent=None, angle=0, \*\*args*)

Bases: `optimeed.visualize.graphs.pyqtgraph.LabelItem`

GraphicsWidget displaying text. Used mainly as axis labels, titles, etc.

Note: To display text inside a scaled view (ViewBox, PlotWidget, etc) use `TextItem`

**setText** (*text, \*\*args*)

Overwrited to add font-family to options

**class myAxis** (*orientation*)

Bases: `optimeed.visualize.graphs.pyqtgraph.AxisItem`

GraphicsItem showing a single plot axis with ticks, values, and label. Can be configured to fit on any side of a plot, Can automatically synchronize its displayed scale with ViewBox items. Ticks can be extended to draw a grid. If `maxTickLength` is negative, ticks point into the plot.

**update\_label**

**\_updateLabel** ()

Internal method to update the label according to the text

```

get_label_pos ()
    Overwrited to place label closer to the axis

resizeEvent (ev=None)
    Overwrited to place label closer to the axis

set_label_pos (orientation, x_offset=0, y_offset=0)

set_number_ticks (number)

```

**traceVisual**

## Module Contents

### Classes

### Functions

### Attributes

**default\_colormap**

**\_normalize\_colors** (*z*)

**class TraceVisual** (*theData, theWGPlot, highlight\_last*)

Bases: `PyQt5.QtCore.QObject`

Defines a trace in a graph.

**class \_ModifiedPaintElem**

Hidden class to manage brushes or pens

**add\_modified\_paintElem** (*index, newPaintElem*)

**modify\_paintElems** (*paintElemsIn\_List*)

Apply transformation to paintElemsIn\_List.

**Param** paintElemsIn\_List: list of brushes or pens to modify

**Returns** False if nothing has been modified, True is something has been modified

**reset\_paintElem** (*index*)

Remove transformation of point index

**reset** ()

**signal\_must\_update**

**hide\_points** ()

Hide all the points

**get\_color** ()

Get colour of the trace, return tuple (r,g,b)

**set\_color** (*color*)

Set colour of the trace, argument as tuple (r,g,b)

**get\_base\_symbol\_brush** ()

Get symbol brush configured for this trace, return *pg.QBrush*

**get\_base\_pen** ()

Get pen configured for this trace, return *pg.QPen*

**get\_base\_symbol\_pen()**

Get symbol pen configured for this trace, return 'pg.QPen'

**get\_base\_symbol()**

Get base symbol configured for this trace, return str of the symbol (e.g. 'o')

**get\_symbol(size)**

Get actual symbols for the trace. If the symbols have been modified: return a list which maps each points to a symbol. Otherwise: return :meth:TraceVisual.get\_base\_symbol()

**updateTrace()**

Forces the trace to refresh.

**get\_length()**

Return number of data to plot

**hide()**

Hides the trace

**show()**

Shows the trace

**toggle(boolean)**

Toggle the trace (hide/show)

**get\_data()**

Get data to plot Data

**get\_brushes(size)**

Get actual brushes for the trace (=symbol filling). return a list which maps each points to a symbol brush

**set\_brush(indexPoint, newbrush, update=True)**

Set the symbol brush for a specific point:

**Parameters**

- **indexPoint** – Index of the point (in the graph) to modify
- **newbrush** – either QBrush or tuple (r, g, b) of the new brush
- **update** – if True, update the trace afterwards. This is slow operation.

**set\_symbol(indexPoint, newSymbol, update=True)**

Set the symbol shape for a specific point:

**Parameters**

- **indexPoint** – Index of the point (in the graph) to modify
- **newSymbol** – string of the new symbol (e.g.: 'o')
- **update** – if True, update the trace afterwards. This is slow operation.

**set\_brushes(list\_indexPoint, list\_newbrush, update=True)**

Same as [set\\_brush\(\)](#) but by taking a list as input

**reset\_brush(indexPoint, update=True)**

Reset the brush of the point indexpoint

**reset\_brushes(list\_indexPoint, update=True)**

Same as [reset\\_brush\(\)](#) but by taking a list as input

**reset\_all\_brushes(update=True)**

Reset all the brushes

**reset\_symbol** (*indexPoint*, *update=True*)

Reset the symbol shape of the point indexpoint

**get\_symbolPens** (*size*)

Get actual symbol pens for the trace (=symbol outline). return a list which maps each points to a symbol pen

**set\_symbolPen** (*indexPoint*, *newPen*, *update=True*)

Set the symbol shape for a specific point:

#### Parameters

- **indexPoint** – Index of the point (in the graph) to modify
- **newPen** – QPen item or tuple of the color (r,g,b)
- **update** – if True, update the trace afterwards. This is slow operation.

**set\_symbolPens** (*list\_indexPoint*, *list\_newpens*, *update=True*)

Same as *set\_symbolPen()* but by taking a list as input

**reset\_symbolPen** (*indexPoint*, *update=True*)

Reset the symbol pen of the point indexpoint

**reset\_symbolPens** (*list\_indexPoint*, *update=True*)

Same as *reset\_symbolPen()* but by taking a list as input

**reset\_all\_symbolPens** (*update=True*)

Reset all the symbol pens

**get\_point** (*indexPoint*)

Return object pyqtgraph.SpotItem

## widget\_graphsVisual

### Module Contents

#### Classes

**class Widget\_graphsVisualLite** (*theGraphs*, *\*\*kwargs*)

Bases: `PyQt5.QtWidgets.QWidget`

Widget element to draw a graph. The traces and graphs to draw are defined in `Graphs` taken as argument. This widget is linked to the excellent third-party library `pyqtgraph`, under MIT license

**signal\_must\_update**

**signal\_graph\_changed**

**set\_graph\_disposition** (*indexGraph*, *row=1*, *col=1*, *rowspan=1*, *colspan=1*)

Change the graphs disposition.

#### Parameters

- **indexGraph** – index of the graph to change
- **row** – row where to place the graph
- **col** – column where to place the graph
- **rowspan** – number of rows across which the graph spans
- **colspan** – number of columns across which the graph spans

**Returns****\_\_create\_graph** (*idGraph*)**\_\_check\_graphs** ()**on\_click** (*plotDataItem*, *clicked\_points*)**update\_graphs** (*singleUpdate=True*)

This method is used to update the graph. This is fast but NOT safe (especially when working with threads). To limit the risks, please use `self.signal_must_update.emit()` instead.

**Parameters** **singleUpdate** – if set to False, the graph will periodically refres each `self.refreshTime`

**fast\_update** ()

Use this method to update the graph in a fast way. NOT THREAD SAFE.

**select\_folder\_and\_export** ()**exportGraphs** (*filename*)

Export the graphs

**export\_txt** (*filename\_txt*)**export\_svg** (*filename*)**export\_tikz** (*foldername\_tikz*)**link\_axes** ()**get\_graph** (*idGraph*) → `optimeed.visualize.graphs.graphVisual.GraphVisual`

Get corresponding `GraphVisual` of the graph `idGraph`

**get\_trace** (*idGraph*, *idTrace*) → `optimeed.visualize.graphs.traceVisual.TraceVisual`

Get corresponding `Tracevisual`

**keyPressEvent** (*event*)

What happens if a key is pressed. R: reset the axes to their default value

**delete\_graph** (*idGraph*)

Delete the graph `idGraph`

**delete** ()**get\_all\_graphsVisual** ()

Return a dictionary {`idGraph`: `GraphVisual`}.

**get\_layout\_buttons** ()

Get the `QGraphicsLayout` where it's possible to add buttons, etc.

**set\_actionOnClick** (*theActionOnClick*)

Action to perform when the graph is clicked

**Parameters** **theActionOnClick** – `on_graph_click_interface`

**Returns****set\_title** (*idGraph*, *titleName*, *\*\*kwargs*)

Set title of the graph

**Parameters**

- **idGraph** – id of the graph
- **titleName** – title to set

```
class Widget_graphsVisual (*args, **kwargs)
```

Bases: *Widget\_graphsVisualLite*

Create a gui for pyqtgraph with trace selection options, export and action on clic choices

```
refreshTraceList ()
```

Refresh all the traces

```
set_actions_on_click (actions)
```

## Package Contents

### Classes

```
class Widget_graphsVisualLite (theGraphs, **kwargs)
```

Bases: `PyQt5.QtWidgets.QWidget`

Widget element to draw a graph. The traces and graphs to draw are defined in `Graphs` taken as argument. This widget is linked to the excellent third-party library `pyqtgraph`, under MIT license

```
signal_must_update
```

```
signal_graph_changed
```

```
set_graph_disposition (indexGraph, row=1, col=1, rowspan=1, colspan=1)
```

Change the graphs disposition.

#### Parameters

- **indexGraph** – index of the graph to change
- **row** – row where to place the graph
- **col** – column where to place the graph
- **rowspan** – number of rows across which the graph spans
- **colspan** – number of columns across which the graph spans

#### Returns

```
__create_graph (idGraph)
```

```
__check_graphs ()
```

```
on_click (plotDataItem, clicked_points)
```

```
update_graphs (singleUpdate=True)
```

This method is used to update the graph. This is fast but NOT safe (especially when working with threads).

To limit the risks, please use `self.signal_must_update.emit()` instead.

**Parameters** **singleUpdate** – if set to False, the graph will periodically refres each `self.refreshTime`

```
fast_update ()
```

Use this method to update the graph in a fast way. NOT THREAD SAFE.

```
select_folder_and_export ()
```

```
exportGraphs (filename)
```

Export the graphs

```
export_txt (filename_txt)
```

```
export_svg (filename)
```

**export\_tikz** (*foldername\_tikz*)

**link\_axes** ()

**get\_graph** (*idGraph*) → optimeed.visualize.graphs.graphVisual.GraphVisual  
Get corresponding GraphVisual of the graph *idGraph*

**get\_trace** (*idGraph*, *idTrace*) → optimeed.visualize.graphs.traceVisual.TraceVisual  
Get corresponding Tracevisual

**keyPressEvent** (*event*)  
What happens if a key is pressed. R: reset the axes to their default value

**delete\_graph** (*idGraph*)  
Delete the graph *idGraph*

**delete** ()

**get\_all\_graphsVisual** ()  
Return a dictionary {*idGraph*: GraphVisual}.

**get\_layout\_buttons** ()  
Get the QGraphicsLayout where it's possible to add buttons, etc.

**set\_actionOnClick** (*theActionOnClick*)  
Action to perform when the graph is clicked

**Parameters** **theActionOnClick** – on\_graph\_click\_interface

**Returns**

**set\_title** (*idGraph*, *titleName*, *\*\*kwargs*)  
Set title of the graph

**Parameters**

- **idGraph** – id of the graph
- **titleName** – title to set

**class Widget\_graphsVisual** (*\*args*, *\*\*kwargs*)  
Bases: *Widget\_graphsVisualLite*

Create a gui for pyqtgraph with trace selection options, export and action on clic choices

**refreshTraceList** ()  
Refresh all the traces

**set\_actions\_on\_click** (*actions*)

## onclick

## animationGUI

## Module Contents

### Classes

**class \_AnimationTrace** (*elements\_list*, *theTrace*)  
Contains all the element to animate for a trace



```

class AnimationElement (elements)

    get ()

    get_element_animations (itemNumber, index_in_show)
        Get the element to show :param itemNumber: item number (0 if only one think to draw) :param index_in_show: index in the list :return: The element to draw

    show_all ()

    delete_all ()

    get_indices_to_show ()

    add_element (indexPoint)

    add_index_to_show (index)

    _remove_index_from_show (index)

    set_curr_brush (index_in_show)

    set_idle_brush (index_in_show)

    get_number_of_elements ()

    map_index (index_in_show)

    get_base_pen ()

class AnimationGUI (id=0, window_title='Animation')
    Bases: PyQt5.QtWidgets.QMainWindow

    Spawns a gui that includes button to create animations nicely when paired with widget_graphs_visual

    SLIDER_MAXIMUM_VALUE = 500

    SLIDER_MINIMUM_VALUE = 1

    add_trace (trace_id, element_list, theTrace)
        Add a trace to the animation.

        Parameters
            • trace_id – id of the trace
            • element_list – List of elements to save: [[OpenGL_item1, text_item1], [OpenGL_item2, text_item2], ... [OpenGL_itemN, text_itemN]]
            • theTrace – TraceVisual

        Returns

    add_elementToTrace (trace_id, indexPoint)

    delete_point (trace_id, thePoint)

    reset_all ()

    delete_all ()

    pause_play ()

    show_all ()

    next_frame ()

    slider_handler ()

```

```
frame_selector()  
set_refreshTime()  
is_empty()  
run()  
closeEvent(_)  
contains_trace(trace_id)  
export_picture()
```

`animation_examples`

## Module Contents

### Classes

**class `Animate_openGL`** (*theOpenGLWidget, theId=0, window\_title='Animation'*)

Bases: `optimeed.visualize.onclick.animationGUI.AnimationGUI`

Implements `DataAnimationVisuals` to show opengl drawing

**update\_widget\_w\_animation** (*key, index, the\_data\_animation*)

What to do when a new element has to be animated. Example:

`self.theOpenGLWidget.set_deviceToDraw(the_data_animation.get_element_animations(0, index))`

#### Parameters

- **key** – key of the trace that has to be animated
- **index** – index that has to be animated
- **the\_data\_animation** – `DataAnimationTrace` that has to be animated

**export\_widget** (*painter*)

Render scene with a painter

Parameters **painter** – `PyQt` painter

**delete\_key\_widgets** (*key*)

What to do when a key has to be deleted

Parameters **key** – key of the trace that has to be deleted

**class `Animate_openGL_and_text`** (*\*args, is\_light=True, \*\*kwargs*)

Bases: `Animate_openGL`

Implements `DataAnimationVisuals` to show opengl drawing and text

**update\_widget\_w\_animation** (*key, index, the\_data\_animation*)

What to do when a new element has to be animated. Example:

`self.theOpenGLWidget.set_deviceToDraw(the_data_animation.get_element_animations(0, index))`

#### Parameters

- **key** – key of the trace that has to be animated
- **index** – index that has to be animated
- **the\_data\_animation** – `DataAnimationTrace` that has to be animated

**get\_interesting\_elements** (*devices\_list*)

Function called upon new trace creation. From a list, takes the interesting elements for animation :param element\_list: :return: new\_element\_list

**class Animate\_lines** (*get\_lines\_method, is\_light=True, theId=0, window\_title='Animation'*)

Bases: *optimeed.visualize.onclick.animationGUI.AnimationGUI*

Implements DataAnimationVisuals to show drawing made out of lines (widget\_line\_drawer)

**export\_widget** (*painter*)

Render scene with a painter

**Parameters painter** – PyQt painter

**delete\_key\_widgets** (*key*)

What to do when a key has to be deleted

**Parameters key** – key of the trace that has to be deleted

**update\_widget\_w\_animation** (*key, index, the\_data\_animation*)

What to do when a new element has to be animated. Example:  
self.theOpenGLWidget.set\_deviceToDraw(the\_data\_animation.get\_element\_animations(0, index))

**Parameters**

- **key** – key of the trace that has to be animated
- **index** – index that has to be animated
- **the\_data\_animation** – DataAnimationTrace that has to be animated

**get\_interesting\_elements** (*devices\_list*)

Function called upon new trace creation. From a list, takes the interesting elements for animation :param element\_list: :return: new\_element\_list

**class Animate\_lines\_and\_text** (*\*args, \*\*kwargs*)

Bases: *Animate\_lines*

Same as DataAnimationLines but also with text

**update\_widget\_w\_animation** (*key, index, the\_data\_animation*)

What to do when a new element has to be animated. Example:  
self.theOpenGLWidget.set\_deviceToDraw(the\_data\_animation.get\_element\_animations(0, index))

**Parameters**

- **key** – key of the trace that has to be animated
- **index** – index that has to be animated
- **the\_data\_animation** – DataAnimationTrace that has to be animated

**collectionExporterGUI**

## Module Contents

### Classes

**class CollectionExporterGUI**

Bases: *PyQt5.QtWidgets.QMainWindow*

Simple gui that allows to export data

```
signal_has_exported
signal_has_reset
exportCollection ()
    Export the collection
reset ()
add_data_to_collection (data)
    Add data to the collection to export
    Parameters data – Whichever type you like
set_collection (theCollection)
```

**onclickInterface**

## Module Contents

### Classes

```
class OnclickInterface
    Interface class for the action to perform when a point is clicked
```

**onclick\_animate**

## Module Contents

### Classes

```
class Onclick_animate (theLinkDataGraph, theAnimation)
    Bases: optimeed.visualize.onclick.onclickInterface.OnclickInterface
    On click: add or remove an element to animate
    graph_clicked (theGraphVisual, index_graph, index_trace, indices_points)
        Action to perform when a graph is clicked
        Parameters
            • theGraphsVisual – class widget_graphs_visual that has called the method
            • index_graph – Index of the graph that has been clicked
            • index_trace – Index of the trace that has been clicked
            • indices_points – graph Indices of the points that have been clicked
        Returns
    get_name ()
```

`onclick_changeSymbol`

## Module Contents

### Classes

**class** `OnClick_changeSymbol` (*theLinkDataGraph*)

Bases: `optimeed.visualize.onclick.onclickInterface.OnclickInterface`

On Click: Change the symbol of the point that is clicked

**graph\_clicked** (*theGraphVisual, index\_graph, index\_trace, indices\_points*)

Action to perform when a graph is clicked

#### Parameters

- **theGraphsVisual** – class `widget_graphs_visual` that has called the method
- **index\_graph** – Index of the graph that has been clicked
- **index\_trace** – Index of the trace that has been clicked
- **indices\_points** – graph Indices of the points that have been clicked

#### Returns

`get_name()`

`onclick_copySomething`

## Module Contents

### Classes

**class** `OnClick_copySomething` (*theDataLink, functionStrFromDevice*)

Bases: `optimeed.visualize.onclick.onclickInterface.OnclickInterface`

On Click: copy something

**graph\_clicked** (*the\_graph\_visual, index\_graph, index\_trace, indices\_points*)

Action to perform when a graph is clicked

#### Parameters

- **theGraphsVisual** – class `widget_graphs_visual` that has called the method
- **index\_graph** – Index of the graph that has been clicked
- **index\_trace** – Index of the trace that has been clicked
- **indices\_points** – graph Indices of the points that have been clicked

#### Returns

`get_name()`

`onclick_delete`

## Module Contents

### Classes

**class** `OnClick_delete` (*theDataLink*)

Bases: `optimeed.visualize.onclick.onclickInterface.OnclickInterface`

On Click: Delete the points from the graph

**graph\_clicked** (*\_theGraphVisual, index\_graph, index\_trace, indices\_points*)

Action to perform when a graph is clicked

#### Parameters

- **theGraphsVisual** – class `widget_graphs_visual` that has called the method
- **index\_graph** – Index of the graph that has been clicked
- **index\_trace** – Index of the trace that has been clicked
- **indices\_points** – graph Indices of the points that have been clicked

#### Returns

`get_name()`

`onclick_exportCollection`

## Module Contents

### Classes

**class** `OnClick_exportCollection` (*theDataLink*)

Bases: `optimeed.visualize.onclick.onclickInterface.OnclickInterface`

On click: export the selected points

**graph\_clicked** (*theGraphVisual, index\_graph, index\_trace, indices\_points*)

Action to perform when a graph is clicked

#### Parameters

- **theGraphsVisual** – class `widget_graphs_visual` that has called the method
- **index\_graph** – Index of the graph that has been clicked
- **index\_trace** – Index of the trace that has been clicked
- **indices\_points** – graph Indices of the points that have been clicked

#### Returns

`reset_graph()`

`get_name()`

`onclick_exportToTxt`

## Module Contents

### Classes

**class** `OnClick_exportToTxt` (*theDataLink, attributes\_shadow=None*)

Bases: `optimeed.visualize.onclick.onclickInterface.OnclickInterface`

On click: export the data of the whole the trace selected

**graph\_clicked** (*theGraphVisual, index\_graph, index\_trace, indices\_points*)

Action to perform when a graph is clicked

#### Parameters

- **theGraphsVisual** – class widget\_graphs\_visual that has called the method
- **index\_graph** – Index of the graph that has been clicked
- **index\_trace** – Index of the trace that has been clicked
- **indices\_points** – graph Indices of the points that have been clicked

#### Returns

`get_name()`

`onclick_exportTrace`

## Module Contents

### Classes

**class** `OnClick_exportTrace` (*theDataLink, getShadow=True*)

Bases: `optimeed.visualize.onclick.onclickInterface.OnclickInterface`

On click: export the data of the whole the trace selected

**graph\_clicked** (*theGraphVisual, index\_graph, index\_trace, indices\_points*)

Action to perform when a graph is clicked

#### Parameters

- **theGraphsVisual** – class widget\_graphs\_visual that has called the method
- **index\_graph** – Index of the graph that has been clicked
- **index\_trace** – Index of the trace that has been clicked
- **indices\_points** – graph Indices of the points that have been clicked

#### Returns

`get_name()`

`onclick_extractPareto`

## Module Contents

### Classes

**class** `OnClick_extractPareto` (*theDataLink, max\_x=False, max\_y=False*)

Bases: `optimeed.visualize.onclick.onclickInterface.OnclickInterface`

On click: extract the pareto from the cloud of points

**graph\_clicked** (*the\_graph\_visual, index\_graph, index\_trace, \_*)

Action to perform when a graph is clicked

#### Parameters

- **theGraphsVisual** – class `widget_graphs_visual` that has called the method
- **index\_graph** – Index of the graph that has been clicked
- **index\_trace** – Index of the trace that has been clicked
- **indices\_points** – graph Indices of the points that have been clicked

#### Returns

`get_name()`

`onclick_measure`

## Module Contents

### Classes

**class** `_LineItem` (*point1, point2*)

Bases: `optimeed.visualize.graphs.pyqtgraph.GraphicsObject`

**Bases:** `GraphicsItem, QtWidgets.QGraphicsObject`

Extension of `QGraphicsObject` with some useful methods (provided by `GraphicsItem`)

**paint** (*p, \*args*)

**boundingRect** ()

**class** `OnClick_measure`

Bases: `optimeed.visualize.onclick.onclickInterface.OnclickInterface`

On Click: Measure distance. Click on two points to perform that action

**graph\_clicked** (*the\_graph\_visual, index\_graph, index\_trace, indices\_points*)

Action to perform when a graph is clicked

#### Parameters

- **theGraphsVisual** – class `widget_graphs_visual` that has called the method
- **index\_graph** – Index of the graph that has been clicked
- **index\_trace** – Index of the trace that has been clicked
- **indices\_points** – graph Indices of the points that have been clicked



**Returns**

```

reset_distance ()
display_distance ()
get_name ()

```

```
onclick_removeTrace
```

**Module Contents****Classes**

```
class Onclick_removeTrace (theDataLink)
```

Bases: *optimeed.visualize.onclick.onclickInterface.OnclickInterface*

Interface class for the action to perform when a point is clicked

```
graph_clicked (theGraphVisual, index_graph, index_trace, _)
```

Action to perform when a graph is clicked

**Parameters**

- **theGraphsVisual** – class widget\_graphs\_visual that has called the method
- **index\_graph** – Index of the graph that has been clicked
- **index\_trace** – Index of the trace that has been clicked
- **indices\_points** – graph Indices of the points that have been clicked

**Returns**

```
get_name ()
```

```
onclick_representDevice
```

**Module Contents****Classes**

```
class RepresentDeviceInterface
```

```
class Onclick_representDevice (theLinkDataGraph, visuals)
```

Bases: *optimeed.visualize.onclick.onclickInterface.OnclickInterface*

On click: show informations about the points (loop through attributes)

```
class DataInformationVisuals
```

```
delete_visual (theVisual)
```

```
add_visual (theVisual, theTrace, indexPoint)
```

```
get_new_index ()
```

```
curr_index ()
```

**graph\_clicked** (*theGraphVisual, index\_graph, index\_trace, indices\_points*)

Action to perform when a point in the graph has been clicked: Creates new window displaying the device and its informations

**get\_name** ()

**onclick\_tojson**

## Module Contents

### Classes

**class Onclick\_tojson** (*theDataLink*)

Bases: *optimeed.visualize.onclick.onclickInterface.OnclickInterface*

Interface class for the action to perform when a point is clicked

**graph\_clicked** (*theGraphVisual, index\_graph, index\_trace, indices\_points*)

Action to perform when a graph is clicked

#### Parameters

- **theGraphsVisual** – class widget\_graphs\_visual that has called the method
- **index\_graph** – Index of the graph that has been clicked
- **index\_trace** – Index of the trace that has been clicked
- **indices\_points** – graph Indices of the points that have been clicked

#### Returns

**get\_name** ()

**representDevice\_examples**

## Module Contents

### Classes

**class Represent\_lines** (*attribute\_lines*)

Bases: *optimeed.visualize.onclick.onclick\_representDevice.RepresentDeviceInterface*

**get\_widget** (*theNewDevice*)

Get Qt widget that represents the device

**Parameters theDevice** – the Device to be represented

**Returns** Qt widget

**class Represent\_txt\_function** (*is\_light=True, convertToHtml=True*)

Bases: *optimeed.visualize.onclick.onclick\_representDevice.RepresentDeviceInterface*

**getTxt** (*theNewDevice*)

**get\_widget** (*theNewDevice*)

Get Qt widget that represents the device

**Parameters** **theDevice** – the Device to be represented

**Returns** Qt widget

```
class Represent_brut_attributes (is_light=True, convertToHtml=True, recursion_level=5)
    Bases: optimeed.visualize.onclick.onclick_representDevice.
    RepresentDeviceInterface
```

**get\_widget** (*theNewDevice*)

Get Qt widget that represents the device

**Parameters** **theDevice** – the Device to be represented

**Returns** Qt widget

```
class Represent_opengl (DeviceDrawer)
    Bases: optimeed.visualize.onclick.onclick_representDevice.
    RepresentDeviceInterface
```

**get\_widget** (*theNewDevice*)

Get Qt widget that represents the device

**Parameters** **theDevice** – the Device to be represented

**Returns** Qt widget

```
class Represent_image (get_base_64_from_device)
    Bases: optimeed.visualize.onclick.onclick_representDevice.
    RepresentDeviceInterface
```

**get\_widget** (*theNewDevice*)

Get Qt widget that represents the device

**Parameters** **theDevice** – the Device to be represented

**Returns** Qt widget

## Package Contents

### Classes

```
class RepresentDeviceInterface
```

```
class Onclick_animate (theLinkDataGraph, theAnimation)
```

Bases: *optimeed.visualize.onclick.onclickInterface.OnclickInterface*

On click: add or remove an element to animate

**graph\_clicked** (*theGraphVisual, index\_graph, index\_trace, indices\_points*)

Action to perform when a graph is clicked

**Parameters**

- **theGraphsVisual** – class widget\_graphs\_visual that has called the method
- **index\_graph** – Index of the graph that has been clicked
- **index\_trace** – Index of the trace that has been clicked
- **indices\_points** – graph Indices of the points that have been clicked

**Returns**

**get\_name** ()

**class Onclick\_changeSymbol** (*theLinkDataGraph*)

Bases: *optimeed.visualize.onclick.onclickInterface.OnclickInterface*

On Click: Change the symbol of the point that is clicked

**graph\_clicked** (*theGraphVisual, index\_graph, index\_trace, indices\_points*)

Action to perform when a graph is clicked

**Parameters**

- **theGraphsVisual** – class widget\_graphs\_visual that has called the method
- **index\_graph** – Index of the graph that has been clicked
- **index\_trace** – Index of the trace that has been clicked
- **indices\_points** – graph Indices of the points that have been clicked

**Returns**

**get\_name** ()

**class Onclick\_copySomething** (*theDataLink, functionStrFromDevice*)

Bases: *optimeed.visualize.onclick.onclickInterface.OnclickInterface*

On Click: copy something

**graph\_clicked** (*the\_graph\_visual, index\_graph, index\_trace, indices\_points*)

Action to perform when a graph is clicked

**Parameters**

- **theGraphsVisual** – class widget\_graphs\_visual that has called the method
- **index\_graph** – Index of the graph that has been clicked
- **index\_trace** – Index of the trace that has been clicked
- **indices\_points** – graph Indices of the points that have been clicked

**Returns**

**get\_name** ()

**class Onclick\_delete** (*theDataLink*)

Bases: *optimeed.visualize.onclick.onclickInterface.OnclickInterface*

On Click: Delete the points from the graph

**graph\_clicked** (*\_theGraphVisual, index\_graph, index\_trace, indices\_points*)

Action to perform when a graph is clicked

**Parameters**

- **theGraphsVisual** – class widget\_graphs\_visual that has called the method
- **index\_graph** – Index of the graph that has been clicked
- **index\_trace** – Index of the trace that has been clicked
- **indices\_points** – graph Indices of the points that have been clicked

**Returns**

**get\_name** ()

**class Onclick\_exportCollection** (*theDataLink*)

Bases: *optimeed.visualize.onclick.onclickInterface.OnclickInterface*

On click: export the selected points

**graph\_clicked** (*theGraphVisual, index\_graph, index\_trace, indices\_points*)

Action to perform when a graph is clicked

#### Parameters

- **theGraphsVisual** – class widget\_graphs\_visual that has called the method
- **index\_graph** – Index of the graph that has been clicked
- **index\_trace** – Index of the trace that has been clicked
- **indices\_points** – graph Indices of the points that have been clicked

#### Returns

**reset\_graph** ()

**get\_name** ()

**class Onclick\_exportToTxt** (*theDataLink, attributes\_shadow=None*)

Bases: *optimeed.visualize.onclick.onclickInterface.OnclickInterface*

On click: export the data of the whole the trace selected

**graph\_clicked** (*theGraphVisual, index\_graph, index\_trace, indices\_points*)

Action to perform when a graph is clicked

#### Parameters

- **theGraphsVisual** – class widget\_graphs\_visual that has called the method
- **index\_graph** – Index of the graph that has been clicked
- **index\_trace** – Index of the trace that has been clicked
- **indices\_points** – graph Indices of the points that have been clicked

#### Returns

**get\_name** ()

**class Onclick\_exportTrace** (*theDataLink, getShadow=True*)

Bases: *optimeed.visualize.onclick.onclickInterface.OnclickInterface*

On click: export the data of the whole the trace selected

**graph\_clicked** (*theGraphVisual, index\_graph, index\_trace, indices\_points*)

Action to perform when a graph is clicked

#### Parameters

- **theGraphsVisual** – class widget\_graphs\_visual that has called the method
- **index\_graph** – Index of the graph that has been clicked
- **index\_trace** – Index of the trace that has been clicked
- **indices\_points** – graph Indices of the points that have been clicked

#### Returns

**get\_name** ()

**class Onclick\_extractPareto** (*theDataLink, max\_x=False, max\_y=False*)

Bases: *optimeed.visualize.onclick.onclickInterface.OnclickInterface*

On click: extract the pareto from the cloud of points

**graph\_clicked** (*the\_graph\_visual, index\_graph, index\_trace, \_*)

Action to perform when a graph is clicked

**Parameters**

- **theGraphsVisual** – class widget\_graphs\_visual that has called the method
- **index\_graph** – Index of the graph that has been clicked
- **index\_trace** – Index of the trace that has been clicked
- **indices\_points** – graph Indices of the points that have been clicked

**Returns**

**get\_name** ()

**class Onclick\_measure**

Bases: *optimeed.visualize.onclick.onclickInterface.OnclickInterface*

On Click: Measure distance. Click on two points to perform that action

**graph\_clicked** (*the\_graph\_visual, index\_graph, index\_trace, indices\_points*)

Action to perform when a graph is clicked

**Parameters**

- **theGraphsVisual** – class widget\_graphs\_visual that has called the method
- **index\_graph** – Index of the graph that has been clicked
- **index\_trace** – Index of the trace that has been clicked
- **indices\_points** – graph Indices of the points that have been clicked

**Returns**

**reset\_distance** ()

**display\_distance** ()

**get\_name** ()

**class Onclick\_removeTrace** (*theDataLink*)

Bases: *optimeed.visualize.onclick.onclickInterface.OnclickInterface*

Interface class for the action to perform when a point is clicked

**graph\_clicked** (*theGraphVisual, index\_graph, index\_trace, \_*)

Action to perform when a graph is clicked

**Parameters**

- **theGraphsVisual** – class widget\_graphs\_visual that has called the method
- **index\_graph** – Index of the graph that has been clicked
- **index\_trace** – Index of the trace that has been clicked
- **indices\_points** – graph Indices of the points that have been clicked

**Returns**

**get\_name** ()

```

class Onclick_representDevice (theLinkDataGraph, visuals)
    Bases: optimeed.visualize.onclick.onclickInterface.OnclickInterface
    On click: show informations about the points (loop through attributes)

    class DataInformationVisuals

        delete_visual (theVisual)
        add_visual (theVisual, theTrace, indexPoint)
        get_new_index ()
        curr_index ()

    graph_clicked (theGraphVisual, index_graph, index_trace, indices_points)
        Action to perform when a point in the graph has been clicked: Creates new window displaying the device
        and its informations

    get_name ()

class Onclick_tojson (theDataLink)
    Bases: optimeed.visualize.onclick.onclickInterface.OnclickInterface
    Interface class for the action to perform when a point is clicked

    graph_clicked (theGraphVisual, index_graph, index_trace, indices_points)
        Action to perform when a graph is clicked

        Parameters
            • theGraphsVisual – class widget_graphs_visual that has called the method
            • index_graph – Index of the graph that has been clicked
            • index_trace – Index of the trace that has been clicked
            • indices_points – graph Indices of the points that have been clicked

        Returns

    get_name ()

class OnclickInterface
    Interface class for the action to perform when a point is clicked

class Represent_opengl (DeviceDrawer)
    Bases: optimeed.visualize.onclick.onclick_representDevice.RepresentDeviceInterface

    get_widget (theNewDevice)
        Get Qt widget that represents the device

        Parameters theDevice – the Device to be represented

        Returns Qt widget

class Represent_image (get_base_64_from_device)
    Bases: optimeed.visualize.onclick.onclick_representDevice.RepresentDeviceInterface

    get_widget (theNewDevice)
        Get Qt widget that represents the device

        Parameters theDevice – the Device to be represented

```

**Returns** Qt widget

**class Represent\_lines** (*attribute\_lines*)

Bases: *optimeed.visualize.onclick.onclick\_representDevice.RepresentDeviceInterface*

**get\_widget** (*theNewDevice*)

Get Qt widget that represents the device

**Parameters** **theDevice** – the Device to be represented

**Returns** Qt widget

**class Represent\_brut\_attributes** (*is\_light=True, convertToHtml=True, recursion\_level=5*)

Bases: *optimeed.visualize.onclick.onclick\_representDevice.RepresentDeviceInterface*

**get\_widget** (*theNewDevice*)

Get Qt widget that represents the device

**Parameters** **theDevice** – the Device to be represented

**Returns** Qt widget

**class Represent\_txt\_function** (*is\_light=True, convertToHtml=True*)

Bases: *optimeed.visualize.onclick.onclick\_representDevice.RepresentDeviceInterface*

**getTxt** (*theNewDevice*)

**get\_widget** (*theNewDevice*)

Get Qt widget that represents the device

**Parameters** **theDevice** – the Device to be represented

**Returns** Qt widget

**class Animate\_lines** (*get\_lines\_method, is\_light=True, theId=0, window\_title='Animation'*)

Bases: *optimeed.visualize.onclick.animationGUI.AnimationGUI*

Implements DataAnimationVisuals to show drawing made out of lines (*widget\_line\_drawer*)

**export\_widget** (*painter*)

Render scene with a painter

**Parameters** **painter** – PyQt painter

**delete\_key\_widgets** (*key*)

What to do when a key has to be deleted

**Parameters** **key** – key of the trace that has to be deleted

**update\_widget\_w\_animation** (*key, index, the\_data\_animation*)

What to do when a new element has to be animated. Example:  
self.theOpenGLWidget.set\_deviceToDraw(the\_data\_animation.get\_element\_animations(0, index))

**Parameters**

- **key** – key of the trace that has to be animated
- **index** – index that has to be animated
- **the\_data\_animation** – DataAnimationTrace that has to be animated



**get\_interesting\_elements** (*devices\_list*)

Function called upon new trace creation. From a list, takes the interesting elements for animation :param element\_list: :return: new\_element\_list

**class Animate OpenGL** (*theOpenGLWidget, theId=0, window\_title='Animation'*)

Bases: *optimeed.visualize.onclick.animationGUI.AnimationGUI*

Implements *DataAnimationVisuals* to show opengl drawing

**update\_widget\_w\_animation** (*key, index, the\_data\_animation*)

What to do when a new element has to be animated. Example:  
self.theOpenGLWidget.set\_deviceToDraw(the\_data\_animation.get\_element\_animations(0, index))

#### Parameters

- **key** – key of the trace that has to be animated
- **index** – index that has to be animated
- **the\_data\_animation** – *DataAnimationTrace* that has to be animated

**export\_widget** (*painter*)

Render scene with a painter

**Parameters painter** – *PyQt* painter

**delete\_key\_widgets** (*key*)

What to do when a key has to be deleted

**Parameters key** – key of the trace that has to be deleted

**class Animate\_lines\_and\_text** (*\*args, \*\*kwargs*)

Bases: *Animate\_lines*

Same as *DataAnimationLines* but also with text

**update\_widget\_w\_animation** (*key, index, the\_data\_animation*)

What to do when a new element has to be animated. Example:  
self.theOpenGLWidget.set\_deviceToDraw(the\_data\_animation.get\_element\_animations(0, index))

#### Parameters

- **key** – key of the trace that has to be animated
- **index** – index that has to be animated
- **the\_data\_animation** – *DataAnimationTrace* that has to be animated

**class Animate OpenGL\_and\_text** (*\*args, is\_light=True, \*\*kwargs*)

Bases: *Animate OpenGL*

Implements *DataAnimationVisuals* to show opengl drawing and text

**update\_widget\_w\_animation** (*key, index, the\_data\_animation*)

What to do when a new element has to be animated. Example:  
self.theOpenGLWidget.set\_deviceToDraw(the\_data\_animation.get\_element\_animations(0, index))

#### Parameters

- **key** – key of the trace that has to be animated
- **index** – index that has to be animated
- **the\_data\_animation** – *DataAnimationTrace* that has to be animated

**get\_interesting\_elements** (*devices\_list*)

Function called upon new trace creation. From a list, takes the interesting elements for animation :param  
element\_list: :return: new\_element\_list

**OpenGL**

**contextHandler**

**Module Contents**

**Classes**

**Attributes**

**MODE\_ZOOM** = 0

**MODE\_ROTATION** = 1

**MODE\_LIGHT** = 2

**NUMBER\_OF\_MODES** = 3

**CLIC\_LEFT** = 0

**CLIC\_RIGHT** = 1

**class SpecialButtonsMapping**

**class MyText** (*color, fontSize, theStr, windowPosition*)

**class ContextHandler**

**set\_specialButtonsMapping** (*theSpecialButtonsMapping*)

**set\_deviceDrawer** (*theDeviceDrawer*)

**set\_deviceToDraw** (*theDeviceToDraw*)

**resizeWindowAction** (*new\_width, new\_height*)

**mouseWheelAction** (*deltaAngle*)

**mouseClicAction** (*button, my\_x, y*)

**mouseMotionAction** (*my\_x, y*)

**keyboardPushAction** (*key*)

**keyboardReleaseAction** (*key, my\_x, y*)

**\_\_draw\_axis\_\_** ()

**redraw** ()

**get\_text\_to\_write** ()

**\_\_lightingInit\_\_** ()

**initialize** ()

**\_\_reset\_\_** ()

deviceDrawerInterface

## Module Contents

### Classes

**class DeviceDrawerInterface**

```

    keyboard_push_action(theKey)
    get_colour_scalebar()
    get_colour_background()
    get_opengl_options()

```

materials

## Module Contents

### Classes

### Attributes

**class MaterialRenderingProperties** (*amb3, dif3, spec3, shin*)

```

    __spec3__ = [0, 0, 0, 0]
    __dif3__ = [0, 0, 0, 0]
    __amb3__ = [0, 0, 0, 0]
    __shin__ = 0
    getSpec3()
    getDif3()
    getAmb3()
    getShin()
    activateMaterialProperties(alpha=1)

```

**Emerald\_material**

**Yellow\_Emerald\_material**

**Brass\_material**

**Bronze\_material**

**Silver\_material**

**Steel\_material**

**Copper\_material**

**Chrome\_material**

**Blue\_material**  
**Red\_material**  
**Green\_material**  
**Cyan\_material**  
**Pink\_material**

**openGL\_library**

## Module Contents

### Functions

**draw\_closedPolygon** (*xClockWise, yClockWise*)  
**draw\_extrudeZ** (*xList, yList, zExtrude*)  
**draw\_triList** (*theTriList*)  
**draw\_lines** (*x, z*)  
**draw\_spiralSheet** (*innerRadius, thickness, length, theAngle, n, reverseDirection=False*)  
**draw\_spiralFront** (*innerRadius, thicknessMaterial, thicknessSpiral, z0, theAngle, n, reverseDirection=False*)  
**draw\_spiralFull** (*innerRadius, outerRadius, thicknessMaterial, thicknessSpiral, length, n*)  
**draw\_spiral** (*innerRadius, outerRadius, thicknessMaterial, thicknessSpiral, length, cutAngle, n*)  
**draw\_simple\_rectangle** (*width, height*)  
**draw\_rectangle** (*rIn, length, thickness, angle, reverseDirection=False*)  
**draw\_2Dring** (*innerRadius, outerRadius, z0, theAngle, n, reverseDirection=False*)  
**draw\_2Dring\_diff\_angle** (*innerRadius, outerRadius, angle\_in, angle\_out, n, reverseDirection=False*)  
**draw\_tubeSheet** (*radius, length, theAngle, n, reverseDirection=False*)  
**draw\_cylinder** (*innerRadius, outerRadius, length, n, translate=0*)  
**draw\_part\_cylinder** (*innerRadius, outerRadius, length, angle, n, translate=0, drawSides=True*)  
**draw\_disk** (*innerRadius, outerRadius, n, translate=0*)  
**draw\_part\_disk** (*innerRadius, outerRadius, thickness, angle, n, translate=0*)  
**draw\_part\_disk\_diff\_angles** (*innerRadius, outerRadius, thickness, angle\_in, angle\_out, n*)  
**draw\_carved\_disk** (*innerRadius, outerRadius, carvedRin, carvedRout, thickness, depth, angle, n, translate=0*)  
**draw\_part\_cylinder\_throat** (*rIn, rOut, rOutThroat, length, lengthThroat, angle, n, translate=0*)  
**drawWireTube** (*diameter, xa, ya, xb, yb, n=50, translateZ=0*)

quaternions

## Module Contents

### Functions

**normalize** (*v*, *tolerance*=0.001)

**q\_mult** (*q1*, *q2*)

**q\_conjugate** (*q*)

**qv\_mult** (*q1*, *v1*)

**axisangle\_to\_q** (*v*, *theta*)

**q\_to\_axisangle** (*q*)

**q\_to\_mat4** (*q*)

triangulate\_polygon

## Module Contents

### Functions

**IsConvex** (*a*, *b*, *c*)

**InTriangle** (*a*, *b*, *c*, *p*)

**IsClockwise** (*poly*)

**GetEar** (*poly*)

**reformatXYtoList** (*xList*, *yList*)

**meshPolygon** (*xList*, *yList*)

## Package Contents

### Classes

### Attributes

**class** DeviceDrawerInterface

**keyboard\_push\_action** (*theKey*)

**get\_colour\_scalebar** ()

**get\_colour\_background** ()

**get\_opengl\_options** ()

**class** MaterialRenderingProperties (*amb3*, *dif3*, *spec3*, *shin*)

```
__spec3__ = [0, 0, 0, 0]
__dif3__ = [0, 0, 0, 0]
__amb3__ = [0, 0, 0, 0]
__shin__ = 0
getSpec3()
getDif3()
getAmb3()
getShin()
activateMaterialProperties(alpha=1)
```

`Emerald_material`

`Yellow_Emerald_material`

`Brass_material`

`Bronze_material`

`Silver_material`

`Steel_material`

`Copper_material`

`Chrome_material`

`Blue_material`

`Red_material`

`Green_material`

`Cyan_material`

`Pink_material`

`selector`

`onselectInterface`

**Module Contents**

**Classes**

`class OnselectInterface`

`onselect_highlight`

**Module Contents**

## Classes

**class Onselect\_highlight** (*theLinkDataGraphs, theWgPlot*)

Bases: *optimeed.visualize.selector.onselectInterface.OnselectInterface*

**selector\_updated** (*selection\_name, the\_collection, selected\_data, not\_selected\_data*)

Action to perform once the data have been selected

### Parameters

- **selection\_name** – name of the selection (deprecated ?)
- **the\_collection** – the collection
- **selected\_data** – indices of the data selected
- **not\_selected\_data** – indices of the data not selected

### Returns

**cancel\_selector** (*selection\_identifier*)

Action to perform when data stopped being selected :param selection\_identifier: identifier that was returned by selector\_updated :return:

**get\_name** ()

Get the name of the action

**Returns** string

**onselect\_newTrace**

## Module Contents

## Classes

**class Onselect\_newTrace** (*theLinkDataGraphs*)

Bases: *optimeed.visualize.selector.onselectInterface.OnselectInterface*

**selector\_updated** (*selection\_name, the\_collection, selected\_data, not\_selected\_data*)

Action to perform once the data have been selected

### Parameters

- **selection\_name** – name of the selection (deprecated ?)
- **the\_collection** – the collection
- **selected\_data** – indices of the data selected
- **not\_selected\_data** – indices of the data not selected

**Returns** identifier that can later be used with cancel\_selector

**cancel\_selector** (*selection\_identifier*)

Action to perform when data stopped being selected :param selection\_identifier: identifier that was returned by selector\_updated :return:

**get\_name** ()

Get the name of the action

**Returns** string

`onselect_splitTrace`

## Module Contents

### Classes

**class** `Onselect_splitTrace` (*theLinkDataGraphs*)

Bases: `optimeed.visualize.selector.onselectInterface.OnselectInterface`

**selector\_updated** (*selection\_name, the\_collection, selected\_data, not\_selected\_data*)

Action to perform once the data have been selected

#### Parameters

- **selection\_name** – name of the selection (deprecated ?)
- **the\_collection** – the collection
- **selected\_data** – indices of the data selected
- **not\_selected\_data** – indices of the data not selected

**Returns** identifier that can later be used with `cancel_selector`

**cancel\_selector** (*selection\_identifiers*)

Action to perform when data stopped being selected :param selection\_identifier: identifier that was returned by selector\_updated :return:

**get\_name** ()

Get the name of the action

**Returns** string

## Package Contents

### Classes

**class** `OnselectInterface`

**class** `Onselect_highlight` (*theLinkDataGraphs, theWgPlot*)

Bases: `optimeed.visualize.selector.onselectInterface.OnselectInterface`

**selector\_updated** (*selection\_name, the\_collection, selected\_data, not\_selected\_data*)

Action to perform once the data have been selected

#### Parameters

- **selection\_name** – name of the selection (deprecated ?)
- **the\_collection** – the collection
- **selected\_data** – indices of the data selected
- **not\_selected\_data** – indices of the data not selected

**Returns**

**cancel\_selector** (*selection\_identifier*)

Action to perform when data stopped being selected :param selection\_identifier: identifier that was returned by selector\_updated :return:



**get\_name()**  
Get the name of the action

**Returns** string

**class Onselect\_newTrace** (*theLinkDataGraphs*)  
Bases: *optimeed.visualize.selector.onselectInterface.OnselectInterface*

**selector\_updated** (*selection\_name, the\_collection, selected\_data, not\_selected\_data*)  
Action to perform once the data have been selected

**Parameters**

- **selection\_name** – name of the selection (deprecated ?)
- **the\_collection** – the collection
- **selected\_data** – indices of the data selected
- **not\_selected\_data** – indices of the data not selected

**Returns** identifier that can later be used with cancel\_selector

**cancel\_selector** (*selection\_identifier*)  
Action to perform when data stopped being selected :param selection\_identifier: identifier that was returned by selector\_updated :return:

**get\_name()**  
Get the name of the action

**Returns** string

**class Onselect\_splitTrace** (*theLinkDataGraphs*)  
Bases: *optimeed.visualize.selector.onselectInterface.OnselectInterface*

**selector\_updated** (*selection\_name, the\_collection, selected\_data, not\_selected\_data*)  
Action to perform once the data have been selected

**Parameters**

- **selection\_name** – name of the selection (deprecated ?)
- **the\_collection** – the collection
- **selected\_data** – indices of the data selected
- **not\_selected\_data** – indices of the data not selected

**Returns** identifier that can later be used with cancel\_selector

**cancel\_selector** (*selection\_identifiers*)  
Action to perform when data stopped being selected :param selection\_identifier: identifier that was returned by selector\_updated :return:

**get\_name()**  
Get the name of the action

**Returns** string

## widgets

**widget\_doubleSlider**

## Module Contents

### Classes

```
class widget_doubleSlider(decimals=3, *args, **kwargs)
    Bases: PyQt5.QtWidgets.QSlider

    doubleValueChanged
    emitDoubleValueChanged()
    value()
    setMinimum(value)
    setMaximum(value)
    setSingleStep(value)
    singleStep()
    setValue(value)
```

`widget_image`

## Module Contents

### Classes

```
class Widget_image(image_b64)
    Bases: PyQt5.QtWidgets.QLabel

    eventFilter(source, event)
    set_image(image_b64)
        Set new image to widget
```

`widget_lineDrawer`

## Module Contents

### Classes

```
class Widget_lineDrawer(minWinHeight=300, minWinWidth=300, is_light=True)
    Bases: PyQt5.QtWidgets.QWidget

    Widget allowing to display several lines easily

    signal_must_update
    on_update_signal(listOfLines)
    delete_lines(key_id)
        Dele the lines :param key_id: id to delete :return:
```

```

set_lines (listOfLines, key_id=0, pen=None)
    Set the lines to display :param listOfLines: list of [x1, y1, x2, y2] corresponding to lines :param key_id:
    id of the trace :param pen: pen used to draw the lines :return:

paintEvent (event, painter=None)

get_extrema_lines ()

```

**widget\_listWithSearch**

## Module Contents

### Classes

```

class Widget_listWithSearch (*args, **kwargs)
    Bases: PyQt5.QtWidgets.QWidget

    get_index_selected ()

    get_name_selected ()

    set_list (names)

    _filter_list ()

    _iter_items ()

```

**widget\_listWithSearchplugin**

## Module Contents

### Classes

```

class Plugin_listWithSearch (parent=None)
    Bases: PyQt5.QtDesigner.QPyDesignerCustomWidgetPlugin

    initialize (core)

    isInitialized ()

    createWidget (parent)

    name ()

    group ()

    icon ()

    toolTip ()

    whatsThis ()

    isContainer ()

    includeFile ()

```

`widget_menuButton`

## Module Contents

### Classes

```
class Widget_menuButton (theParentButton)
    Bases: PyQt5.QtWidgets.QMenu
    Same as QMenu, but integrates it behind a button more easily.
    showEvent (QShowEvent)
    mouseReleaseEvent (QMouseEvent)
```

`widget_openGL`

## Module Contents

### Classes

```
class Widget_openGL (parent=None)
    Bases: PyQt5.QtWidgets.QOpenGLWidget
    Interface that provides opengl capabilities. Ensures zoom, light, rotation, etc.
    sizeHint ()
    minimumSizeHint ()
    set_deviceDrawer (theDeviceDrawer)
        Set a drawer optimeed.visualize.widgets.openGL.deviceDrawerInterface.
        DeviceDrawerInterface
    set_deviceToDraw (theDeviceToDraw)
        Set the device to draw
    initializeGL ()
    paintGL ()
    resizeGL (w, h)
    mousePressEvent (event)
    mouseMoveEvent (event)
    keyPressEvent (event)
    wheelEvent (QWheelEvent)
```

`widget_tableWithSearch`

## Module Contents

## Classes

```
class Widget_tableWithSearch(*args,**kwargs)
    Bases: PyQt5.QtWidgets.QWidget

    cellChanged

    hideRow(row)

    showRow(row)

    force_hide_row(row)

    remove_forced_hide_row(row)

    get_entries_selected()

    _cellChanged()

    set_entries(names,numColumns=3,hidden=False)

    get_shown_entries()

    set_item(row,col,item)

    get_item(row,col)

    _filter_list()

    _iter_items()
```

widget\_tableWithSearchplugin

## Module Contents

### Classes

```
class Plugin_tableWithSearch(parent=None)
    Bases: PyQt5.QtDesigner.QPyDesignerCustomWidgetPlugin

    initialize(core)

    isInitialized()

    createWidget(parent)

    name()

    group()

    icon()

    toolTip()

    whatsThis()

    isContainer()

    includeFile()
```

widget\_text

## Module Contents

### Classes

```
class Widget_text (theText, is_light=False, convertToHtml=False)
    Bases: PyQt5.QtWidgets.QLabel
    Widget able to display a text

    set_text (theText, convertToHtml=False)
        Set the text to display

class Widget_text_scrollable (theText, is_light=False, convertToHtml=False)
    Bases: PyQt5.QtWidgets.QWidget
    Same as widget_text but scrollable

    set_text (theText, convertToHtml=False)
```

## Package Contents

### Classes

```
class Widget_image (image_b64)
    Bases: PyQt5.QtWidgets.QLabel

    eventFilter (source, event)

    set_image (image_b64)
        Set new image to widget

class Widget_lineDrawer (minWinHeight=300, minWinWidth=300, is_light=True)
    Bases: PyQt5.QtWidgets.QWidget
    Widget allowing to display several lines easily

    signal_must_update

    on_update_signal (listOfLines)

    delete_lines (key_id)
        Delete the lines :param key_id: id to delete :return:

    set_lines (listOfLines, key_id=0, pen=None)
        Set the lines to display :param listOfLines: list of [x1, y1, x2, y2] corresponding to lines :param key_id:
        id of the trace :param pen: pen used to draw the lines :return:

    paintEvent (event, painter=None)

    get_extrema_lines ()

class Widget_listWithSearch (*args, **kwargs)
    Bases: PyQt5.QtWidgets.QWidget

    get_index_selected ()

    get_name_selected ()

    set_list (names)
```

```

    _filter_list ()
    _iter_items ()

class Widget_menuButton (theParentButton)
    Bases: PyQt5.QtWidgets.QMenu

    Same as QMenu, but integrates it behind a button more easily.

    showEvent (QShowEvent)

    mouseReleaseEvent (QMouseEvent)

class Widget_opengl (parent=None)
    Bases: PyQt5.QtWidgets.QOpenGLWidget

    Interface that provides opengl capabilities. Ensures zoom, light, rotation, etc.

    sizeHint ()

    minimumSizeHint ()

    set_deviceDrawer (theDeviceDrawer)
        Set a drawer optimeed.visualize.widgets.opengl.deviceDrawerInterface.
        DeviceDrawerInterface

    set_deviceToDraw (theDeviceToDraw)
        Set the device to draw

    initializeGL ()

    paintGL ()

    resizeGL (w, h)

    mousePressEvent (event)

    mouseMoveEvent (event)

    keyPressEvent (event)

    wheelEvent (QWheelEvent)

class Widget_tableWithSearch (*args, **kwargs)
    Bases: PyQt5.QtWidgets.QWidget

    cellChanged

    hideRow (row)

    showRow (row)

    force_hide_row (row)

    remove_forced_hide_row (row)

    get_entries_selected ()

    _cellChanged ()

    set_entries (names, numColumns=3, hidden=False)

    get_shown_entries ()

    set_item (row, col, item)

    get_item (row, col)

    _filter_list ()

```

```
    _iter_items ()

class Widget_text (theText, is_light=False, convertToHtml=False)
    Bases: PyQt5.QtWidgets.QLabel
    Widget able to display a text

    set_text (theText, convertToHtml=False)
        Set the text to display

class Widget_text_scrollable (theText, is_light=False, convertToHtml=False)
    Bases: PyQt5.QtWidgets.QWidget
    Same as widget_text but scrollable

    set_text (theText, convertToHtml=False)
```

**displayCollections**

## Module Contents

### Classes

### Functions

```
_is_object_selected (object_in, min_max_attributes)

_select_and_apply_action (theCollections, min_max_attributes, theAction, selectionName)

class CollectionDisplay
    Bases: PyQt5.QtWidgets.QMainWindow
    GUI to display a collection.

    add_collection (theCollection, name="")
        Add a collection to the GUI

    set_shadow (master_collectionId, shadow_collection)
        Set a shadow collection to master_collectionID (see DataLink.set_shadow_collection)

    remove_collection (theCollection)
        Remove collection from the GUI

    update_graphs ()

    set_actions_on_click (theActionsOnClick)
        Set actions to be performed when graph is clicked

    get_datalink ()

    _initialize (theCollection)

    _set_x ()

    _set_y ()

    _set_z ()

    set_action_selector (theAction)

    _selector_to ()

    _remove_item_selector ()
```



```

_cancel_selector()
_apply_selector()
_reset_colors()

```

displayOptimization

## Module Contents

### Classes

### Functions

```

check_if_must_plot(elem)

run_optimization_displayer(*args, **kwargs)

class OptimizationDisplayer(theOptiParameters, theOptiHistoric, additionalWidgets=None,
                             light_background=False)
    Bases: optimeed.core.Option_class
    Class used to display optimization process in real time

    signal_optimization_over

    SHOW_CONSTRAINTS = 0

    set_actionsOnClick(theList)
        Set actions to perform on click, list of on_graph_click_interface

    generate_optimizationGraphs()
        Generates the optimization graphs. :return: Graphs, LinkDataGraph,
        :class: '~optimeed.visulaize.gui.widgets.widget_graphs_visual.widget_graphs_visual

    __change_appearance_violate_constraints()

    __refresh()

    start_autorefresh(timer_autosave)

    stop_autorefresh()

    __set_graphs_disposition()
        Set nicely the graphs disposition

    launch_optimization(args_opti, kwargs_opti, refresh_time=0.1,
                        max_nb_points_convergence=100)
        Perform the optimization and spawn the convergence graphs afterwards. :param args_opti: arguments (as
        list) destined to launch the optimization :param kwargs_opti: keywords arguments (as dict) destined to
        launch the optimization :param refresh_time: float indicating the refresh time of the graphs. If it becomes
        laggy -> use a higher one. :param max_nb_points_convergence: maximum number of points in the graph
        that displays the convergence. Put None if performance is not an issue.

    close_windows()

    display_graphs(theGraphs)

    create_main_window()
        From the widgets and the actions on click, spawn a window and put a gui around widgetsGraphsVisual.

```

`displaySensitivity`

## Module Contents

### Classes

### Functions

**analyse\_sobol\_plot\_convergence** (*theDict*, *sobol*='S1', *title*="", *hold*=True)

Plot convergence of the sobol indices.

#### Parameters

- **theDict** – Dictionary containing sobol indices
- **sobol** – Key of the dictionary to investigate
- **title** – Title of the convergence window
- **hold** – If true, this function will be blocking (otherwise use `start_qt_mainloop`)

**Returns** window containing convergence graphs

**analyse\_sobol\_plot\_indices** (*theSensitivityParameters*: *optimeed.consolidate.SensitivityParameters*,  
*objectives*, *title*="", *hold*=True)

Plot first and total order sobol indices.

#### Parameters

- **theSensitivityParameters** – Parameters used for sensitivity study
- **objectives** – List of evaluated objectives to analyse
- **title** – Title of the window
- **hold** – If true, this function will be blocking (otherwise use `plt.show()`)

**Returns**

**analyse\_sobol\_plot\_2ndOrder\_indices** (*theSensitivityParameters*: *optimeed.consolidate.SensitivityParameters*, *objectives*,  
*title*="", *hold*=True)

Plot second order sobol indices. Args and kwargs are the same as `analyse_sobol_plot_indices`

**class SensitivityDisplayer**

Bases: `PyQt5.QtWidgets.QMainWindow`

GUI to display a sensitivity analysis.

**add\_study** (*theCollection*, *theParameters*, *name*)

Add sensitivity study to the GUI

#### Parameters

- **theCollection** – Results of the sensitivity study
- **theParameters** – Parameters of the sensitivity study
- **name** – Name (for the GUI) of the sensitivity study

**Returns**

`_set_study` (*index*)

`_get_sobol_indices` ()

```

_get_S1_conv ()
_get_ST_conv ()

```

**fastPlot**

## Module Contents

### Classes

### Functions

### Attributes

**class \_PlotHolders**

```

add_plot (x, y, **kwargs)
get_wgGraphs ()
new_plot ()
set_title (theTitle, **kwargs)
reset ()
axis_equal ()

```

**class WindowHolders**

```

set_currFigure (currFigure)
add_plot (*args, **kwargs)
set_title (*args, **kwargs)
new_figure ()
new_plot ()
show ()
get_curr_plotHolder ()
get_wgGraphs (fig=None)
get_all_figures ()
axis_equal ()
add_action_on_click (theAction)

```

**myWindows**

```

plot (x, y, hold=False, **kwargs)
    Plot new trace

```

```

show ()
    Show (start qt mainloop) graphs. Blocking

```

**figure** (*numb=None*)  
Set current figure

**add\_action\_on\_click** (*theAction*)

**set\_title** (*theTitle, \*\*kwargs*)  
Set title of the plot

**axis\_equal** ()

**get\_all\_figures** ()  
Get all existing figures

**get\_wgGraphs** (*fig=None*)  
Advanced option. :return: widget\_graphs\_visual

**fastPlot3**

## Module Contents

### Functions

### Attributes

**hasPlotly** = True

**\_do\_scatterPlot** (*theData: optimeed.core.ScatterPlot3*)

**mainWindow**

## Module Contents

### Classes

**class MainWindow** (*QtWidgetList, isLight=True, actionOnWindowClosed=None, neverCloseWin-*  
*dow=False, title\_window='Awesome Visualisation Tool', size=None*)  
Bases: PyQt5.QtWidgets.QMainWindow

Main class that spawns a Qt window. Use `run()` to display it.

**set\_actionOnClose** (*actionOnWindowClosed*)

**closeEvent** (*event*)

**run** (*hold=False*)  
Display the window

**keyPressEvent** (*event*)

**process\_mainloop**

## Module Contents

### Functions

## Attributes

**app**

**start\_qt\_mainloop()**  
Starts qt mainloop, which is necessary for qt to handle events

**stop\_qt\_mainloop()**  
Stops qt mainloop and resumes to program

**process\_qt\_events()**  
Process current qt events

**viewOptimizationResults**

## Module Contents

### Classes

**class \_OptiProjectLoader** (*foldername, kwargsPlot=None*)  
A loader for an opti project.

**get\_devices()** → optimeed.core.ListDataStruct\_Interface

**get\_logopti()** → optimeed.core.ListDataStruct\_Interface

**get\_convergence()**

**get\_kwargs()**

**get\_nbr\_objectives()**

**class ViewOptimizationResults**  
Convenience class to display the results of an optimization

**add\_opti\_project** (*foldername, kwargsPlot=None*)  
Add an opti project to visualize.

#### Parameters

- **foldername** – the folder containing the saved files. (as string)
- **kwargsPlot** – Check `kwargs ~optimeed.core.graphs.Data`

**get\_data\_link()** → optimeed.core.LinkDataGraph  
Return the object [LinkDataGraph](#)

**display\_graphs** (*theActionsOnClick=None, kwargs\_common=None, keep\_alive=True, max\_nb\_points\_convergence=None, light\_background=False*)  
Generates the optimization graphs.

#### Parameters

- **theActionsOnClick** – list of actions to perform when a graph is clicked
- **kwargs\_common** – plot options (from Data class) to apply to all the graphs (ex: {"is\_scattered": True}).
- **keep\_alive** – if set to true, this method will be blocking. Otherwise you should manually call `start_qt_mainloop()`.

- **max\_nb\_points\_convergence** – maximum number of points in the graph that displays the convergence. Put None if performance is not an issue.
- **light\_background** – boolean, True or False for White or Black background color in graphs

**Returns** widget\_graphs\_visual for the log opti, widget\_graphs\_visual for the convergence (widget\_graphs\_visual)

## Package Contents

### Classes

### Functions

### Attributes

#### **class CollectionDisplayer**

Bases: PyQt5.QtWidgets.QMainWindow

GUI to display a collection.

**add\_collection** (*theCollection*, *name*=")

Add a collection to the GUI

**set\_shadow** (*master\_collectionId*, *shadow\_collection*)

Set a shadow collection to master\_collectionID (see DataLink.set\_shadow\_collection)

**remove\_collection** (*theCollection*)

Remove collection from the GUI

**update\_graphs** ()

**set\_actions\_on\_click** (*theActionsOnClick*)

Set actions to be performed when graph is clicked

**get\_datalink** ()

**\_initialize** (*theCollection*)

**\_set\_x** ()

**\_set\_y** ()

**\_set\_z** ()

**set\_action\_selector** (*theAction*)

**\_selector\_to** ()

**\_remove\_item\_selector** ()

**\_cancel\_selector** ()

**\_apply\_selector** ()

**\_reset\_colors** ()

#### **class SensitivityDisplayer**

Bases: PyQt5.QtWidgets.QMainWindow

GUI to display a sensitivity analysis.

**add\_study** (*theCollection, theParameters, name*)

Add sensitivity study to the GUI

**Parameters**

- **theCollection** – Results of the sensitivity study
- **theParameters** – Parameters of the sensitivity study
- **name** – Name (for the GUI) of the sensitivity study

**Returns**

**\_set\_study** (*index*)

**\_get\_sobol\_indices** ()

**\_get\_S1\_conv** ()

**\_get\_ST\_conv** ()

**analyse\_sobol\_plot\_indices** (*theSensitivityParameters: optimeed.consolidate.SensitivityParameters, objectives, title="", hold=True*)

Plot first and total order sobol indices.

**Parameters**

- **theSensitivityParameters** – Parameters used for sensitivity study
- **objectives** – List of evaluated objectives to analyse
- **title** – Title of the window
- **hold** – If true, this function will be blocking (otherwise use plt.show())

**Returns**

**analyse\_sobol\_plot\_convergence** (*theDict, sobol='S1', title="", hold=True*)

Plot convergence of the sobol indices.

**Parameters**

- **theDict** – Dictionary containing sobol indices
- **sobol** – Key of the dictionary to investigate
- **title** – Title of the convergence window
- **hold** – If true, this function will be blocking (otherwise use start\_qt\_mainloop)

**Returns** window containing convergence graphs

**analyse\_sobol\_plot\_2ndOrder\_indices** (*theSensitivityParameters: optimeed.consolidate.SensitivityParameters, objectives, title="", hold=True*)

Plot second order sobol indices. Args and kwargs are the same as analyse\_sobol\_plot\_indices

**class OptimizationDisplayer** (*theOptiParameters, theOptiHistoric, additionalWidgets=None, light\_background=False*)

Bases: *optimeed.core.Option\_class*

Class used to display optimization process in real time

**signal\_optimization\_over**

**SHOW\_CONSTRAINTS = 0**

**set\_actionsOnClick** (*theList*)

Set actions to perform on click, list of on\_graph\_click\_interface

```
generate_optimizationGraphs ()
    Generates the optimization graphs. :return: Graphs, LinkDataGraph,
    :class:'~optimeed.visulaize.gui.widgets.widget_graphs_visual.widget_graphs_visual

__change_appearance_violate_constraints ()

__refresh ()

start_autorefresh (timer_autosave)

stop_autorefresh ()

__set_graphs_disposition ()
    Set nicely the graphs disposition

launch_optimization (args_opti, kwargs_opti, refresh_time=0.1,
    max_nb_points_convergence=100)
    Perform the optimization and spawn the convergence graphs afterwards. :param args_opti: arguments (as
    list) destined to launch the optimization :param kwargs_opti: keywords arguments (as dict) destined to
    launch the optimization :param refresh_time: float indicating the refresh time of the graphs. If it becomes
    laggy -> use a higher one. :param max_nb_points_convergence: maximum number of points in the graph
    that displays the convergence. Put None if performance is not an issue.

close_windows ()

display_graphs (theGraphs)

create_main_window ()
    From the widgets and the actions on click, spawn a window and put a gui around widgetsGraphsVisual.

class ViewOptimizationResults
    Convenience class to display the results of an optimization

add_opti_project (foldername, kwargsPlot=None)
    Add an opti project to visualize.

    Parameters
    • foldername – the folder containing the saved files. (as string)
    • kwargsPlot – Check kkwargs ~optimeed.core.graphs.Data

get_data_link () → optimeed.core.LinkDataGraph
    Return the object LinkDataGraph

display_graphs (theActionsOnClick=None, kwargs_common=None, keep_alive=True,
    max_nb_points_convergence=None, light_background=False)
    Generates the optimization graphs.

    Parameters
    • theActionsOnClick – list of actions to perform when a graph is clicked
    • kwargs_common – plot options (from Data class) to apply to all the graphs (ex:
    {"is_scattered": True}).
    • keep_alive – if set to true, this method will be blocking. Otherwise you should manu-
    ally call start_qt_mainloop().
    • max_nb_points_convergence – maximum number of points in the graph that dis-
    plays the convergence. Put None if performance is not an issue.
    • light_background – boolean, True or False for White or Black background color in
    graphs
```



**Returns** widget\_graphs\_visual for the log opti, widget\_graphs\_visual for the convergence  
(widget\_graphs\_visual)

**class Widget\_graphsVisual** (\*args, \*\*kwargs)

Bases: *Widget\_graphsVisualLite*

Create a gui for pyqtgraph with trace selection options, export and action on clic choices

**refreshTraceList** ()

Refresh all the traces

**set\_actions\_on\_click** (actions)

**class MainWindow** (QtWidgetList, isLight=True, actionOnWindowClosed=None, neverCloseWin-  
dow=False, title\_window='Awesome Visualisation Tool', size=None)

Bases: PyQt5.QtWidgets.QMainWindow

Main class that spawns a Qt window. Use run () to display it.

**set\_actionOnClose** (actionOnWindowClosed)

**closeEvent** (event)

**run** (hold=False)

Display the window

**keyPressEvent** (event)

**start\_qt\_mainloop** ()

Starts qt mainloop, which is necessary for qt to handle events

**class Data** (x: list, y: list, x\_label="", y\_label="", legend="", is\_scattered=False, transfo\_x=lambda self-  
Data, x: x, transfo\_y=lambda selfData, y: y, xlim=None, ylim=None, permutations=None,  
sort\_output=False, color=None, alpha=255, symbol='o', symbolsize=8, fillsymbol=True, out-  
linesymbol=1.8, linestyle='-', width=2, meta=None)

This class is used to store informations necessary to plot a 2D graph. It has to be combined with a gui to be useful (ex. pyqtgraph)

**set\_kwargs** (kwargs)

Set a kwarg after creation of the class

**set\_data** (x: list, y: list)

Overwrites current datapoints with new set

**set\_meta** (meta)

Set associated 'Z' data

**get\_x** ()

Get x coordinates of datapoints

**get\_symbolsizes** ()

Get size of the symbols

**symbol\_isfilled** ()

Check if symbols has to be filled or not

**get\_symbolOutline** ()

Get color factor of outline of symbols

**get\_length\_data** ()

Get number of points

**get\_xlim** ()

Get x limits of viewbox

**get\_ylim()**  
Get y limits of viewbox

**get\_y()**  
Get y coordinates of datapoints

**get\_meta()**  
Get associated 'Z' data

**get\_color()**  
Get color of the line, without transformation

**get\_color\_alpha()**  
Get color of the line. Return r, g, b in 0, 255 scale

**get\_alpha()**  
Get opacity

**get\_width()**  
Get width of the line

**get\_number\_of\_points()**  
Get number of points

**get\_plot\_data()**  
Call this method to get the x and y coordinates of the points that have to be displayed. => After transformation, and after permutations.  
  
**Returns** x (list), y (list)

**get\_plot\_meta(x, y)**  
Call this method to get the z coordinates of the points that been displayed. => After transformation, and after permutations.  
  
**Returns** z (list)

**get\_permutations(x=None)**  
Return the transformation 'permutation': xplot[i] = xdata[permutation[i]]

**get\_invert\_permutations()**  
Return the inverse of permutations: xdata[i] = xplot[revert[i]]

**get\_dataIndex\_from\_graphIndex(index\_graph\_point)**  
From an index given in graph, recovers the index of the data.  
  
**Parameters** **index\_graph\_point** – Index in the graph  
**Returns** index of the data

**get\_dataIndices\_from\_graphIndices(index\_graph\_point\_list)**  
Same as get\_dataIndex\_from\_graphIndex but with a list in entry. Can (?) improve performances for huge dataset.  
  
**Parameters** **index\_graph\_point\_list** – List of Index in the graph  
**Returns** List of index of the data

**get\_graphIndex\_from\_dataIndex(index\_data)**  
From an index given in the data, recovers the index of the graph.  
  
**Parameters** **index\_data** – Index in the data  
**Returns** index of the graph

**get\_graphIndices\_from\_dataIndices** (*index\_data\_list*)  
 Same as `get_graphIndex_from_dataIndex` but with a list in entry. Can (?) improve performances for huge dataset.

**Parameters** *index\_data\_list* – List of Index in the data

**Returns** List of index of the graph

**set\_permutations** (*permutations*)  
 Set permutations between datapoints of the trace

**Parameters** *permutations* – list of indices to plot (example: [0, 2, 1] means that the first point will be plotted, then the third, then the second one)

**get\_x\_label** ()  
 Get x label of the trace

**get\_y\_label** ()  
 Get y label of the trace

**get\_legend** ()  
 Get name of the trace

**get\_symbol** ()  
 Get symbol

**add\_point** (*x, y*)  
 Add point(s) to trace (inputs can be list or numeral)

**delete\_point** (*index\_point*)  
 Delete a point from the datapoints

**isScattered** ()  
 Check if plot is scatteded

**set\_indices\_points\_to\_plot** (*indices*)  
 Set indices points to plot

**get\_indices\_points\_to\_plot** ()  
 Get indices points to plot

**get\_linestyle** ()  
 Get linestyle

**\_\_str\_\_** ()  
 Return str(self).

**export\_str** ()  
 Method to save the points constituting the trace

**set\_color** (*theColor*)  
 Set trace color

**set\_legend** (*theLegend*)  
 Set legend

**class Graphs**  
 Contains several Graph

**updateChildren** ()

**add\_trace\_firstGraph** (*data, updateChildren=True*)  
 Same as `add_trace`, but only if graphs has only one id :param data: :param updateChildren: :return:

**add\_trace** (*idGraph*, *data*, *updateChildren=True*)

Add a trace to the graph

**Parameters**

- **idGraph** – id of the graph
- **data** – *Data*
- **updateChildren** – Automatically calls callback functions

**Returns** id of the created trace

**remove\_trace** (*idGraph*, *idTrace*, *updateChildren=True*)

Remove the trace from the graph

**Parameters**

- **idGraph** – id of the graph
- **idTrace** – id of the trace to remove
- **updateChildren** – Automatically calls callback functions

**get\_first\_graph** ()

Get id of the first graph

**Returns** id of the first graph

**get\_graph** (*idGraph*)

Get graph object at idgraph

**Parameters** **idGraph** – id of the graph to get

**Returns** *Graph*

**get\_all\_graphs\_ids** ()

Get all ids of the graphs

**Returns** list of id graphs

**get\_all\_graphs** ()

Get all graphs. Return dict {id: *Graph*}

**add\_graph** (*updateChildren=True*)

Add a new graph

**Returns** id of the created graph

**remove\_graph** (*idGraph*)

Delete a graph

**Parameters** **idGraph** – id of the graph to delete

**add\_update\_method** (*childObject*)

Add a callback each time a graph is modified.

**Parameters** **childObject** – method without arguments

**export\_str** ()

Export all the graphs in text

**Returns** *str*

**merge** (*otherGraphs*)

**reset** ()

```

    is_empty()

class Onclick_measure
    Bases: optimeed.visualize.onclick.onclickInterface.OnclickInterface
    On Click: Measure distance. Click on two points to perform that action
    graph_clicked(the_graph_visual, index_graph, index_trace, indices_points)
        Action to perform when a graph is clicked

        Parameters
        • theGraphsVisual – class widget_graphs_visual that has called the method
        • index_graph – Index of the graph that has been clicked
        • index_trace – Index of the trace that has been clicked
        • indices_points – graph Indices of the points that have been clicked

        Returns

    reset_distance()
    display_distance()
    get_name()

class _PlotHolders

    add_plot(x, y, **kwargs)
    get_wgGraphs()
    new_plot()
    set_title(theTitle, **kwargs)
    reset()
    axis_equal()

class WindowHolders

    set_currFigure(currFigure)
    add_plot(*args, **kwargs)
    set_title(*args, **kwargs)
    new_figure()
    new_plot()
    show()

    get_curr_plotHolder()
    get_wgGraphs(fig=None)
    get_all_figures()
    axis_equal()
    add_action_on_click(theAction)

myWindows

```

```
plot (x, y, hold=False, **kwargs)
    Plot new trace

show ()
    Show (start qt mainloop) graphs. Blocking

figure (numb=None)
    Set current figure

add_action_on_click (theAction)

set_title (theTitle, **kwargs)
    Set title of the plot

axis_equal ()

get_all_figures ()
    Get all existing figures

get_wgGraphs (fig=None)
    Advanced option. :return: widget_graphs_visual

class FilledContourPlot (*args, **kwargs)
    Bases: ContourPlot

class ContourPlot (*args, **kwargs)
    Bases: GridPlot_Generic

    get_levels ()

    get_number_of_contours ()

class SurfPlot (X, Y, Z, **kwargs)
    Bases: GridPlot_Generic

class MeshPlot (X, Y, Z, **kwargs)
    Bases: GridPlot_Generic

class ScatterPlot3 (x, y, z, **kwargs)
    Bases: Plot3D_Generic

    get_plot_data ()

    get_color ()

printIfShown (theStr, show_type=SHOW_DEBUG, isToPrint=True, appendTypeName=True, end='n')

SHOW_WARNING = 0

hasPlotly = True

_do_scatterPlot (theData: optimeed.core.ScatterPlot3)

class Widget_graphsVisualLite (theGraphs, **kwargs)
    Bases: PyQt5.QtWidgets.QWidget

    Widget element to draw a graph. The traces and graphs to draw are defined in Graphs taken as argument. This
    widget is linked to the excellent third-party library pyqtgraph, under MIT license

    signal_must_update

    signal_graph_changed

    set_graph_disposition (indexGraph, row=1, col=1, rowspan=1, colspan=1)
        Change the graphs disposition.
```

#### Parameters

- **indexGraph** – index of the graph to change
- **row** – row where to place the graph
- **col** – column where to place the graph
- **rowspan** – number of rows across which the graph spans
- **colspan** – number of columns across which the graph spans

#### Returns

**\_\_create\_graph** (*idGraph*)

**\_\_check\_graphs** ()

**on\_click** (*plotDataItem*, *clicked\_points*)

**update\_graphs** (*singleUpdate=True*)

This method is used to update the graph. This is fast but NOT safe (especially when working with threads). To limit the risks, please use `self.signal_must_update.emit()` instead.

**Parameters singleUpdate** – if set to False, the graph will periodically refres each `self.refreshtime`

**fast\_update** ()

Use this method to update the graph in a fast way. NOT THREAD SAFE.

**select\_folder\_and\_export** ()

**exportGraphs** (*filename*)

Export the graphs

**export\_txt** (*filename\_txt*)

**export\_svg** (*filename*)

**export\_tikz** (*foldername\_tikz*)

**link\_axes** ()

**get\_graph** (*idGraph*) → `optimeed.visualize.graphs.graphVisual.GraphVisual`

Get corresponding `GraphVisual` of the graph `idGraph`

**get\_trace** (*idGraph*, *idTrace*) → `optimeed.visualize.graphs.traceVisual.TraceVisual`

Get corresponding `Tracevisual`

**keyPressEvent** (*event*)

What happens if a key is pressed. R: reset the axes to their default value

**delete\_graph** (*idGraph*)

Delete the graph `idGraph`

**delete** ()

**get\_all\_graphsVisual** ()

Return a dictionary {`idGraph`: `GraphVisual`}.

**get\_layout\_buttons** ()

Get the `QGraphicsLayout` where it's possible to add buttons, etc.

**set\_actionOnClick** (*theActionOnClick*)

Action to perform when the graph is clicked

**Parameters theActionOnClick** – `on_graph_click_interface`

#### Returns

**set\_title** (*idGraph*, *titleName*, *\*\*kwargs*)  
Set title of the graph

**Parameters**

- **idGraph** – id of the graph
- **titleName** – title to set

**class Widget\_graphsVisual** (*\*args*, *\*\*kwargs*)  
Bases: *Widget\_graphsVisualLite*

Create a gui for pyqtgraph with trace selection options, export and action on clic choices

**refreshTraceList** ()  
Refresh all the traces

**set\_actions\_on\_click** (*actions*)

**class Onclick\_representDevice** (*theLinkDataGraph*, *visuals*)  
Bases: *optimeed.visualize.onclick.onclickInterface.OnclickInterface*

On click: show informations about the points (loop through attributes)

**class DataInformationVisuals**

**delete\_visual** (*theVisual*)

**add\_visual** (*theVisual*, *theTrace*, *indexPoint*)

**get\_new\_index** ()

**curr\_index** ()

**graph\_clicked** (*theGraphVisual*, *index\_graph*, *index\_trace*, *indices\_points*)  
Action to perform when a point in the graph has been clicked: Creates new window displaying the device and its informations

**get\_name** ()

**class RepresentDeviceInterface**

**class Onclick\_animate** (*theLinkDataGraph*, *theAnimation*)  
Bases: *optimeed.visualize.onclick.onclickInterface.OnclickInterface*

On click: add or remove an element to animate

**graph\_clicked** (*theGraphVisual*, *index\_graph*, *index\_trace*, *indices\_points*)  
Action to perform when a graph is clicked

**Parameters**

- **theGraphsVisual** – class widget\_graphs\_visual that has called the method
- **index\_graph** – Index of the graph that has been clicked
- **index\_trace** – Index of the trace that has been clicked
- **indices\_points** – graph Indices of the points that have been clicked

**Returns**

**get\_name** ()

**class Onclick\_changeSymbol** (*theLinkDataGraph*)  
Bases: *optimeed.visualize.onclick.onclickInterface.OnclickInterface*



On Click: Change the symbol of the point that is clicked

**graph\_clicked** (*theGraphVisual, index\_graph, index\_trace, indices\_points*)

Action to perform when a graph is clicked

#### Parameters

- **theGraphsVisual** – class widget\_graphs\_visual that has called the method
- **index\_graph** – Index of the graph that has been clicked
- **index\_trace** – Index of the trace that has been clicked
- **indices\_points** – graph Indices of the points that have been clicked

#### Returns

**get\_name** ()

**class Onclick\_copySomething** (*theDataLink, functionStrFromDevice*)

Bases: *optimeed.visualize.onclick.onclickInterface.OnclickInterface*

On Click: copy something

**graph\_clicked** (*the\_graph\_visual, index\_graph, index\_trace, indices\_points*)

Action to perform when a graph is clicked

#### Parameters

- **theGraphsVisual** – class widget\_graphs\_visual that has called the method
- **index\_graph** – Index of the graph that has been clicked
- **index\_trace** – Index of the trace that has been clicked
- **indices\_points** – graph Indices of the points that have been clicked

#### Returns

**get\_name** ()

**class Onclick\_delete** (*theDataLink*)

Bases: *optimeed.visualize.onclick.onclickInterface.OnclickInterface*

On Click: Delete the points from the graph

**graph\_clicked** (*\_theGraphVisual, index\_graph, index\_trace, indices\_points*)

Action to perform when a graph is clicked

#### Parameters

- **theGraphsVisual** – class widget\_graphs\_visual that has called the method
- **index\_graph** – Index of the graph that has been clicked
- **index\_trace** – Index of the trace that has been clicked
- **indices\_points** – graph Indices of the points that have been clicked

#### Returns

**get\_name** ()

**class Onclick\_exportCollection** (*theDataLink*)

Bases: *optimeed.visualize.onclick.onclickInterface.OnclickInterface*

On click: export the selected points

**graph\_clicked** (*theGraphVisual, index\_graph, index\_trace, indices\_points*)

Action to perform when a graph is clicked

**Parameters**

- **theGraphsVisual** – class widget\_graphs\_visual that has called the method
- **index\_graph** – Index of the graph that has been clicked
- **index\_trace** – Index of the trace that has been clicked
- **indices\_points** – graph Indices of the points that have been clicked

**Returns**

**reset\_graph** ()

**get\_name** ()

**class Onclick\_exportToTxt** (*theDataLink, attributes\_shadow=None*)

Bases: *optimeed.visualize.onclick.onclickInterface.OnclickInterface*

On click: export the data of the whole the trace selected

**graph\_clicked** (*theGraphVisual, index\_graph, index\_trace, indices\_points*)

Action to perform when a graph is clicked

**Parameters**

- **theGraphsVisual** – class widget\_graphs\_visual that has called the method
- **index\_graph** – Index of the graph that has been clicked
- **index\_trace** – Index of the trace that has been clicked
- **indices\_points** – graph Indices of the points that have been clicked

**Returns**

**get\_name** ()

**class Onclick\_exportTrace** (*theDataLink, getShadow=True*)

Bases: *optimeed.visualize.onclick.onclickInterface.OnclickInterface*

On click: export the data of the whole the trace selected

**graph\_clicked** (*theGraphVisual, index\_graph, index\_trace, indices\_points*)

Action to perform when a graph is clicked

**Parameters**

- **theGraphsVisual** – class widget\_graphs\_visual that has called the method
- **index\_graph** – Index of the graph that has been clicked
- **index\_trace** – Index of the trace that has been clicked
- **indices\_points** – graph Indices of the points that have been clicked

**Returns**

**get\_name** ()

**class Onclick\_extractPareto** (*theDataLink, max\_x=False, max\_y=False*)

Bases: *optimeed.visualize.onclick.onclickInterface.OnclickInterface*

On click: extract the pareto from the cloud of points

**graph\_clicked** (*the\_graph\_visual, index\_graph, index\_trace, \_*)

Action to perform when a graph is clicked

#### Parameters

- **theGraphsVisual** – class widget\_graphs\_visual that has called the method
- **index\_graph** – Index of the graph that has been clicked
- **index\_trace** – Index of the trace that has been clicked
- **indices\_points** – graph Indices of the points that have been clicked

#### Returns

**get\_name** ()

**class Onclick\_measure**

Bases: *optimeed.visualize.onclick.onclickInterface.OnclickInterface*

On Click: Measure distance. Click on two points to perform that action

**graph\_clicked** (*the\_graph\_visual, index\_graph, index\_trace, indices\_points*)

Action to perform when a graph is clicked

#### Parameters

- **theGraphsVisual** – class widget\_graphs\_visual that has called the method
- **index\_graph** – Index of the graph that has been clicked
- **index\_trace** – Index of the trace that has been clicked
- **indices\_points** – graph Indices of the points that have been clicked

#### Returns

**reset\_distance** ()

**display\_distance** ()

**get\_name** ()

**class Onclick\_removeTrace** (*theDataLink*)

Bases: *optimeed.visualize.onclick.onclickInterface.OnclickInterface*

Interface class for the action to perform when a point is clicked

**graph\_clicked** (*theGraphVisual, index\_graph, index\_trace, \_*)

Action to perform when a graph is clicked

#### Parameters

- **theGraphsVisual** – class widget\_graphs\_visual that has called the method
- **index\_graph** – Index of the graph that has been clicked
- **index\_trace** – Index of the trace that has been clicked
- **indices\_points** – graph Indices of the points that have been clicked

#### Returns

**get\_name** ()

**class Onclick\_tojson** (*theDataLink*)

Bases: *optimeed.visualize.onclick.onclickInterface.OnclickInterface*

Interface class for the action to perform when a point is clicked

**graph\_clicked** (*theGraphVisual, index\_graph, index\_trace, indices\_points*)

Action to perform when a graph is clicked

**Parameters**

- **theGraphsVisual** – class widget\_graphs\_visual that has called the method
- **index\_graph** – Index of the graph that has been clicked
- **index\_trace** – Index of the trace that has been clicked
- **indices\_points** – graph Indices of the points that have been clicked

**Returns**

**get\_name** ()

**class OnclickInterface**

Interface class for the action to perform when a point is clicked

**class Represent\_opengl** (*DeviceDrawer*)

Bases: *optimeed.visualize.onclick.onclick\_representDevice.  
RepresentDeviceInterface*

**get\_widget** (*theNewDevice*)

Get Qt widget that represents the device

**Parameters theDevice** – the Device to be represented

**Returns** Qt widget

**class Represent\_image** (*get\_base\_64\_from\_device*)

Bases: *optimeed.visualize.onclick.onclick\_representDevice.  
RepresentDeviceInterface*

**get\_widget** (*theNewDevice*)

Get Qt widget that represents the device

**Parameters theDevice** – the Device to be represented

**Returns** Qt widget

**class Represent\_lines** (*attribute\_lines*)

Bases: *optimeed.visualize.onclick.onclick\_representDevice.  
RepresentDeviceInterface*

**get\_widget** (*theNewDevice*)

Get Qt widget that represents the device

**Parameters theDevice** – the Device to be represented

**Returns** Qt widget

**class Represent\_brut\_attributes** (*is\_light=True, convertToHtml=True, recursion\_level=5*)

Bases: *optimeed.visualize.onclick.onclick\_representDevice.  
RepresentDeviceInterface*

**get\_widget** (*theNewDevice*)

Get Qt widget that represents the device

**Parameters theDevice** – the Device to be represented

**Returns** Qt widget

```

class Represent_txt_function (is_light=True, convertToHtml=True)
    Bases: optimeed.visualize.onclick.onclick_representDevice.RepresentDeviceInterface

    getTxt (theNewDevice)

    get_widget (theNewDevice)
        Get Qt widget that represents the device

        Parameters theDevice – the Device to be represented

        Returns Qt widget

class Animate_lines (get_lines_method, is_light=True, theId=0, window_title='Animation')
    Bases: optimeed.visualize.onclick.animationGUI.AnimationGUI

    Implements DataAnimationVisuals to show drawing made out of lines (widget_line_drawer)

    export_widget (painter)
        Render scene with a painter

        Parameters painter – PyQt painter

    delete_key_widgets (key)
        What to do when a key has to be deleted

        Parameters key – key of the trace that has to be deleted

    update_widget_w_animation (key, index, the_data_animation)
        What to do when a new element has to be animated. Example:
        self.theOpenGLWidget.set_deviceToDraw(the_data_animation.get_element_animations(0, index))

        Parameters
        • key – key of the trace that has to be animated
        • index – index that has to be animated
        • the_data_animation – DataAnimationTrace that has to be animated

    get_interesting_elements (devices_list)
        Function called upon new trace creation. From a list, takes the interesting elements for animation :param
        element_list: :return: new_element_list

class Animate_openGL (theOpenGLWidget, theId=0, window_title='Animation')
    Bases: optimeed.visualize.onclick.animationGUI.AnimationGUI

    Implements DataAnimationVisuals to show opengl drawing

    update_widget_w_animation (key, index, the_data_animation)
        What to do when a new element has to be animated. Example:
        self.theOpenGLWidget.set_deviceToDraw(the_data_animation.get_element_animations(0, index))

        Parameters
        • key – key of the trace that has to be animated
        • index – index that has to be animated
        • the_data_animation – DataAnimationTrace that has to be animated

    export_widget (painter)
        Render scene with a painter

        Parameters painter – PyQt painter

```

**delete\_key\_widgets** (*key*)

What to do when a key has to be deleted

**Parameters** **key** – key of the trace that has to be deleted

**class Animate\_lines\_and\_text** (*\*args, \*\*kwargs*)

Bases: *Animate\_lines*

Same as *DataAnimationLines* but also with text

**update\_widget\_w\_animation** (*key, index, the\_data\_animation*)

What to do when a new element has to be animated. Example:

`self.theOpenGLWidget.set_deviceToDraw(the_data_animation.get_element_animations(0, index))`

**Parameters**

- **key** – key of the trace that has to be animated
- **index** – index that has to be animated
- **the\_data\_animation** – *DataAnimationTrace* that has to be animated

**class Animate\_openGL\_and\_text** (*\*args, is\_light=True, \*\*kwargs*)

Bases: *Animate\_openGL*

Implements *DataAnimationVisuals* to show opengl drawing and text

**update\_widget\_w\_animation** (*key, index, the\_data\_animation*)

What to do when a new element has to be animated. Example:

`self.theOpenGLWidget.set_deviceToDraw(the_data_animation.get_element_animations(0, index))`

**Parameters**

- **key** – key of the trace that has to be animated
- **index** – index that has to be animated
- **the\_data\_animation** – *DataAnimationTrace* that has to be animated

**get\_interesting\_elements** (*devices\_list*)

Function called upon new trace creation. From a list, takes the interesting elements for animation :param  
element\_list: :return: new\_element\_list

**class DeviceDrawerInterface**

**keyboard\_push\_action** (*theKey*)

**get\_colour\_scalebar** ()

**get\_colour\_background** ()

**get\_opengl\_options** ()

**class MaterialRenderingProperties** (*amb3, dif3, spec3, shin*)

**\_\_spec3\_\_** = [0, 0, 0, 0]

**\_\_dif3\_\_** = [0, 0, 0, 0]

**\_\_amb3\_\_** = [0, 0, 0, 0]

**\_\_shin\_\_** = 0

**getSpec3** ()

**getDif3** ()

```

    getAmb3()
    getShin()
    activateMaterialProperties(alpha=1)
Emerald_material
Yellow_Emerald_material
Brass_material
Bronze_material
Silver_material
Steel_material
Copper_material
Chrome_material
Blue_material
Red_material
Green_material
Cyan_material
Pink_material
class OnselectInterface
class Onselect_highlight(theLinkDataGraphs, theWgPlot)
    Bases: optimeed.visualize.selector.onselectInterface.OnselectInterface
    selector_updated(selection_name, the_collection, selected_data, not_selected_data)
        Action to perform once the data have been selected

        Parameters
            • selection_name – name of the selection (deprecated ?)
            • the_collection – the collection
            • selected_data – indices of the data selected
            • not_selected_data – indices of the data not selected

        Returns

    cancel_selector(selection_identifier)
        Action to perform when data stopped being selected :param selection_identifier: identifier that was returned by selector_updated :return:

    get_name()
        Get the name of the action

        Returns string

class Onselect_newTrace(theLinkDataGraphs)
    Bases: optimeed.visualize.selector.onselectInterface.OnselectInterface
    selector_updated(selection_name, the_collection, selected_data, not_selected_data)
        Action to perform once the data have been selected

        Parameters

```

- **selection\_name** – name of the selection (deprecated ?)
- **the\_collection** – the collection
- **selected\_data** – indices of the data selected
- **not\_selected\_data** – indices of the data not selected

**Returns** identifier that can later be used with `cancel_selector`

**cancel\_selector** (*selection\_identifier*)

Action to perform when data stopped being selected :param selection\_identifier: identifier that was returned by selector\_updated :return:

**get\_name** ()

Get the name of the action

**Returns** string

**class Onselect\_splitTrace** (*theLinkDataGraphs*)

Bases: `optimeed.visualize.selector.onselectInterface.OnselectInterface`

**selector\_updated** (*selection\_name, the\_collection, selected\_data, not\_selected\_data*)

Action to perform once the data have been selected

**Parameters**

- **selection\_name** – name of the selection (deprecated ?)
- **the\_collection** – the collection
- **selected\_data** – indices of the data selected
- **not\_selected\_data** – indices of the data not selected

**Returns** identifier that can later be used with `cancel_selector`

**cancel\_selector** (*selection\_identifiers*)

Action to perform when data stopped being selected :param selection\_identifier: identifier that was returned by selector\_updated :return:

**get\_name** ()

Get the name of the action

**Returns** string

**class Widget\_listWithSearch** (*\*args, \*\*kwargs*)

Bases: `PyQt5.QtWidgets.QWidget`

**get\_index\_selected** ()

**get\_name\_selected** ()

**set\_list** (*names*)

**\_filter\_list** ()

**\_iter\_items** ()

**class Widget\_image** (*image\_b64*)

Bases: `PyQt5.QtWidgets.QLabel`

**eventFilter** (*source, event*)

**set\_image** (*image\_b64*)

Set new image to widget



```

class Widget_lineDrawer (minWinHeight=300, minWinWidth=300, is_light=True)
    Bases: PyQt5.QtWidgets.QWidget

    Widget allowing to display several lines easily

    signal_must_update

    on_update_signal (listOfLines)

    delete_lines (key_id)
        Delete the lines :param key_id: id to delete :return:

    set_lines (listOfLines, key_id=0, pen=None)
        Set the lines to display :param listOfLines: list of [x1, y1, x2, y2] corresponding to lines :param key_id:
        id of the trace :param pen: pen used to draw the lines :return:

    paintEvent (event, painter=None)

    get_extrema_lines ()

class Widget_menuButton (theParentButton)
    Bases: PyQt5.QtWidgets.QMenu

    Same as QMenu, but integrates it behind a button more easily.

    showEvent (QShowEvent)

    mouseReleaseEvent (QMouseEvent)

class Widget_openGL (parent=None)
    Bases: PyQt5.QtWidgets.QOpenGLWidget

    Interface that provides opengl capabilities. Ensures zoom, light, rotation, etc.

    sizeHint ()

    minimumSizeHint ()

    set_deviceDrawer (theDeviceDrawer)
        Set a drawer optimeed.visualize.widgets.openGL.deviceDrawerInterface.
        DeviceDrawerInterface

    set_deviceToDraw (theDeviceToDraw)
        Set the device to draw

    initializeGL ()

    paintGL ()

    resizeGL (w, h)

    mousePressEvent (event)

    mouseMoveEvent (event)

    keyPressEvent (event)

    wheelEvent (QWheelEvent)

class Widget_tableWithSearch (*args, **kwargs)
    Bases: PyQt5.QtWidgets.QWidget

    cellChanged

    hideRow (row)

    showRow (row)

```

```
force_hide_row (row)
remove_forced_hide_row (row)
get_entries_selected ()
_cellChanged ()
set_entries (names, numColumns=3, hidden=False)
get_shown_entries ()
set_item (row, col, item)
get_item (row, col)
_filter_list ()
_iter_items ()

class Widget_text (theText, is_light=False, convertToHtml=False)
    Bases: PyQt5.QtWidgets.QLabel
    Widget able to display a text

    set_text (theText, convertToHtml=False)
        Set the text to display

class Widget_text_scrollable (theText, is_light=False, convertToHtml=False)
    Bases: PyQt5.QtWidgets.QWidget
    Same as widget_text but scrollable

    set_text (theText, convertToHtml=False)
```

## 6.1.2 Package Contents

VERSION = 2.1.1

## 7.1 Developer documentation

### 7.1.1 Packages for doc:

- *pip install sphinx*
- *pip install sphinx-autoapi*
- *pip install sphinx\_rtd\_theme*

### 7.1.2 To regenerate API:

- uncomment line # 'autoapi.extension' in conf.py.
- run make html
- run hack.py script
- recomment line # 'autoapi.extension'
- run make html
- Eventually update project on <https://readthedocs.org/projects/optimeed/>

### 7.1.3 To updata packages on PyPi:

- Change version in setup.py and in optimeed/\_\_init\_\_.py
- Create new wheel file code::*python setup.py sdist bdist\_wheel*
- Upload it on pypi code::*twine upload dist/\**



---

## Python Module Index

---

### O

optimeed, 19

optimeed consolidate, 19

optimeed consolidate.fit, 19

optimeed consolidate.parametric\_analysis, 20

optimeed consolidate.sensitivity\_analysis, 21

optimeed consolidate.sensitivity\_analysis\_evaluation, 23

optimeed core, 27

optimeed core.additional\_tools, 30

optimeed core.ansi2html, 27

optimeed core.ansi2html.converter, 27

optimeed core.ansi2html.style, 29

optimeed core.ansi2html.util, 30

optimeed core.collection, 32

optimeed core.color\_palette, 35

optimeed core.commonImport, 35

optimeed core.graphs, 36

optimeed core.graphs3, 40

optimeed core.inkscape\_manager, 41

optimeed core.linkDataGraph, 41

optimeed core.myjson, 43

optimeed core.options, 44

optimeed core.tikzTranslator, 46

optimeed core.tools, 47

optimeed optimize, 66

optimeed optimize.characterization, 67

optimeed optimize.characterization.characterization, 67

optimeed optimize.characterization.interfaceCharacterization, 67

optimeed optimize.mathsToPhysics, 68

optimeed optimize.mathsToPhysics.interfaceMathsToPhysics, 68

optimeed optimize.mathsToPhysics.mathsToPhysics, 68

optimeed optimize.objAndCons, 69

optimeed optimize.objAndCons.fastObjCons, 69

optimeed optimize.objAndCons.interfaceObjCons, 70

optimeed optimize.optiAlgorithms, 70

optimeed optimize.optiAlgorithms.algorithmInterface, 76

optimeed optimize.optiAlgorithms.convergence, 70

optimeed optimize.optiAlgorithms.convergence.evolution, 70

optimeed optimize.optiAlgorithms.convergence.hyper, 71

optimeed optimize.optiAlgorithms.convergence.inter, 72

optimeed optimize.optiAlgorithms.monobjective\_PSO, 76

optimeed optimize.optiAlgorithms.multiObjective\_GA, 77

optimeed optimize.optiAlgorithms.NLOpt\_Algorithm, 75

optimeed optimize.optiAlgorithms.pyswarm, 73

optimeed optimize.optiAlgorithms.pyswarm.pso, 73

optimeed optimize.optiHistoric, 81

optimeed optimize.optimizer, 84

optimeed optimize.optiVariable, 82

optimeed visualize, 90

optimeed visualize.displayCollections, 132

optimeed visualize.displayOptimization, 133

optimeed visualize.displaySensitivity, 134

optimeed visualize.fastPlot, 135

optimeed visualize.fastPlot3, 136

optimeed visualize.graphs, 90

optimeed visualize.graphs.colormap\_pyqtgraph, 90

[optimeed.visualize.graphs.graphVisual](#), 90  
[optimeed.visualize.graphs.pyqtgraphRedefine](#), 92  
[optimeed.visualize.graphs.traceVisual](#), 95  
[optimeed.visualize.graphs.widget\\_graphsViewport](#), 97  
[optimeed.visualize.mainWindow](#), 136  
[optimeed.visualize.onclick](#), 100  
[optimeed.visualize.onclick.animation\\_examples](#), 102  
[optimeed.visualize.onclick.animationGUI](#), 100  
[optimeed.visualize.onclick.collectionExportGUI](#), 103  
[optimeed.visualize.onclick.onclick\\_animate](#), 104  
[optimeed.visualize.onclick.onclick\\_changeSymbol](#), 105  
[optimeed.visualize.onclick.onclick\\_copySomething](#), 105  
[optimeed.visualize.onclick.onclick\\_delete](#), 106  
[optimeed.visualize.onclick.onclick\\_exportCollection](#), 106  
[optimeed.visualize.onclick.onclick\\_exportToTxt](#), 107  
[optimeed.visualize.onclick.onclick\\_exportTrace](#), 107  
[optimeed.visualize.onclick.onclick\\_extractParenthesis](#), 108  
[optimeed.visualize.onclick.onclick\\_measure](#), 108  
[optimeed.visualize.onclick.onclick\\_removeTrace](#), 109  
[optimeed.visualize.onclick.onclick\\_representDevice](#), 109  
[optimeed.visualize.onclick.onclick\\_tojson](#), 110  
[optimeed.visualize.onclick.onclickInterface](#), 104  
[optimeed.visualize.onclick.representDevice\\_examples](#), 110  
[optimeed.visualize.openGL](#), 118  
[optimeed.visualize.openGL.contextHandler](#), 118  
[optimeed.visualize.openGL.deviceDrawerInterface](#), 119  
[optimeed.visualize.openGL.materials](#), 119  
[optimeed.visualize.openGL.openGL\\_library](#), 120  
[optimeed.visualize.openGL.quaternions](#), 121  
[optimeed.visualize.openGL.triangulate\\_polygon](#), 121  
[optimeed.visualize.process\\_mainloop](#), 136  
[optimeed.visualize.selector](#), 122  
[optimeed.visualize.selector.onselect\\_highlight](#), 122  
[optimeed.visualize.selector.onselect\\_newTrace](#), 123  
[optimeed.visualize.selector.onselect\\_splitTrace](#), 124  
[optimeed.visualize.selector.onselectInterface](#), 122  
[optimeed.visualize.viewOptimizationResults](#), 137  
[optimeed.visualize.widgets](#), 125  
[optimeed.visualize.widgets.widget\\_doubleSlider](#), 125  
[optimeed.visualize.widgets.widget\\_image](#), 126  
[optimeed.visualize.widgets.widget\\_lineDrawer](#), 126  
[optimeed.visualize.widgets.widget\\_listWithSearch](#), 127  
[optimeed.visualize.widgets.widget\\_listWithSearchplus](#), 127  
[optimeed.visualize.widgets.widget\\_menuButton](#), 127  
[optimeed.visualize.widgets.widget\\_openGL](#), 128  
[optimeed.visualize.widgets.widget\\_tableWithSearch](#), 128  
[optimeed.visualize.widgets.widget\\_tableWithSearchplus](#), 128  
[optimeed.visualize.widgets.widget\\_text](#), 129

## Symbols

\_AnimationTrace (class in *optimeed.visualize.onclick.animationGUI*), 100  
 \_AnimationTrace.AnimationElement (class in *optimeed.visualize.onclick.animationGUI*), 100  
 \_COMPRESS\_SAVE\_STR (*ListDataStruct* attribute), 23, 33, 53  
 \_COMPRESS\_SAVE\_STR (*Performance\_ListDataStruct* attribute), 34, 54  
 \_DATA\_STR (*ListDataStruct* attribute), 23, 33, 53  
 \_Device (class in *optimeed consolidate.fit*), 19  
 \_Evaluator (class in *optimeed.optimize.optimizer*), 84  
 \_LineItem (class in *optimeed.visualize.onclick.onclick\_measure*), 108  
 \_NBR\_ELEMENTS (*Performance\_ListDataStruct* attribute), 34, 54  
 \_Objective (class in *optimeed consolidate.fit*), 19  
 \_OptiProjectLoader (class in *optimeed.visualize.viewOptimizationResults*), 137  
 \_PlotHolders (class in *optimeed.visualize*), 145  
 \_PlotHolders (class in *optimeed.visualize.fastPlot*), 135  
 \_STACK\_SIZE (*Performance\_ListDataStruct* attribute), 34, 54  
 \_State (class in *optimeed.core.ansi2html.converter*), 28  
 \_\_amb3\_\_ (*MaterialRenderingProperties* attribute), 119, 122, 154  
 \_\_author\_\_ (in module *optimeed.optimize.optiAlgorithms.convergence.hypervolume*), 71  
 \_\_change\_appearance\_violate\_constraints\_\_ (*OptimizationDisplayer* method), 133, 140  
 \_\_check\_graphs\_\_ () (*Widget\_graphsVisualLite* method), 98, 99, 147  
 \_\_create\_graph\_\_ () (*Widget\_graphsVisualLite* method), 98, 99, 147  
 \_\_dif3\_\_ (*MaterialRenderingProperties* attribute), 119, 122, 154  
 \_\_draw\_axis\_\_ () (*ContextHandler* method), 118  
 \_\_getstate\_\_ () (*AutosaveStruct* method), 25, 33, 53  
 \_\_len\_\_ () (*ListDataStruct* method), 23, 33, 53  
 \_\_len\_\_ () (*MultiList* method), 72  
 \_\_lightingInit\_\_ () (*ContextHandler* method), 118  
 \_\_refresh\_\_ () (*OptimizationDisplayer* method), 133, 140  
 \_\_reset\_\_ () (*ContextHandler* method), 118  
 \_\_set\_graphs\_disposition\_\_ () (*OptimizationDisplayer* method), 133, 140  
 \_\_setstate\_\_ () (*AutosaveStruct* method), 25, 33, 53  
 \_\_shin\_\_ (*MaterialRenderingProperties* attribute), 119, 122, 154  
 \_\_spec3\_\_ (*MaterialRenderingProperties* attribute), 119, 121, 154  
 \_\_str\_\_ () (*AutosaveStruct* method), 24, 32, 53  
 \_\_str\_\_ () (*Binary\_OptimizationVariable* method), 83, 88  
 \_\_str\_\_ () (*Data* method), 38, 58, 143  
 \_\_str\_\_ () (*DataStruct\_Interface* method), 32, 53  
 \_\_str\_\_ () (*HowToPlotGraph* method), 41, 60  
 \_\_str\_\_ () (*Integer\_OptimizationVariable* method), 83, 88  
 \_\_str\_\_ () (*InterfaceCharacterization* method), 67, 85  
 \_\_str\_\_ () (*InterfaceObjCons* method), 70, 86  
 \_\_str\_\_ () (*MathsToPhysics* method), 68, 69, 85  
 \_\_str\_\_ () (*Monobjective\_PSO* method), 77, 81, 87  
 \_\_str\_\_ () (*MultiList* method), 72  
 \_\_str\_\_ () (*MultiList.Node* method), 72  
 \_\_str\_\_ () (*MultiObjective\_GA* method), 79, 80, 87  
 \_\_str\_\_ () (*NLOpt\_Algorithm* method), 76  
 \_\_str\_\_ () (*OptimizationVariable* method), 82  
 \_\_str\_\_ () (*Option\_class* method), 23, 45, 63  
 \_\_str\_\_ () (*Real\_OptimizationVariable* method), 83, 88  
 \_\_str\_\_ () (*Rule* method), 29  
 \_\_apply\_regex\_\_ () (*Ansi2HTMLConverter* method), 29, 30

- `_apply_selector()` (*CollectionDisplay*er method), 133, 138
- `_cancel_selector()` (*CollectionDisplay*er method), 132, 138
- `_cellChanged()` (*Widget\_tableWithSearch* method), 129, 131, 158
- `_collapse_cursor()` (*Ansi2HTMLConverter* method), 29, 30
- `_do_scatterPlot()` (in module *optimeed.visualize*), 146
- `_do_scatterPlot()` (in module *optimeed.visualize.fastPlot3*), 136
- `_extract_N_steps()` (*EvolutionaryConvergence* method), 71, 73
- `_filename_sensitivityparams` (in module *optimeed consolidate.sensitivity\_analysis*), 21
- `_filename_sensitivityresults` (in module *optimeed consolidate.sensitivity\_analysis*), 21
- `_filter_list()` (*Widget\_listWithSearch* method), 127, 130, 156
- `_filter_list()` (*Widget\_tableWithSearch* method), 129, 131, 158
- `_find_class()` (in module *optimeed.core*), 65
- `_find_class()` (in module *optimeed.core.myjson*), 43
- `_find_missings()` (in module *optimeed consolidate.sensitivity\_analysis*), 22
- `_foldername_embarrassingly_parallel_results` (in module *optimeed consolidate.sensitivity\_analysis*), 21
- `_format_data_lines()` (*ListDataStruct* method), 24, 33, 54
- `_format_fx_fs()` (in module *optimeed.optimize.optiAlgorithms.pyswarm*), 74
- `_format_fx_fs()` (in module *optimeed.optimize.optiAlgorithms.pyswarm.pso*), 73
- `_format_str_save()` (*ListDataStruct* method), 24, 33, 54
- `_get_S1_conv()` (*SensitivityDisplay*er method), 134, 139
- `_get_ST_conv()` (*SensitivityDisplay*er method), 135, 139
- `_get_annotations()` (in module *optimeed.core*), 65
- `_get_annotations()` (in module *optimeed.core.myjson*), 44
- `_get_attributes_to_save()` (in module *optimeed.core*), 65
- `_get_attributes_to_save()` (in module *optimeed.core.myjson*), 44
- `_get_job_args()` (in module *optimeed consolidate.sensitivity\_analysis*), 22
- `_get_json_module_tree()` (*ListDataStruct* method), 24, 33, 54
- `_get_json_str_at_index()` (*Performance\_ListDataStruct* method), 34, 55
- `_get_list_from_file()` (*Performance\_ListDataStruct* method), 34, 54
- `_get_object_class()` (in module *optimeed.core*), 65
- `_get_object_class()` (in module *optimeed.core.myjson*), 43
- `_get_object_module()` (in module *optimeed.core*), 65
- `_get_object_module()` (in module *optimeed.core.myjson*), 43
- `_get_sensitivity_result()` (in module *optimeed consolidate.sensitivity\_analysis*), 22
- `_get_sobol_indices()` (*SensitivityDisplay*er method), 134, 139
- `_get_str_mainfile()` (*Performance\_ListDataStruct* method), 34, 54
- `_html_template` (in module *optimeed.core.ansi2html.converter*), 28
- `_initialize()` (*CollectionDisplay*er method), 132, 138
- `_initialize()` (*Performance\_ListDataStruct* method), 34, 54
- `_instantiates_annotated_object()` (in module *optimeed.core*), 65
- `_instantiates_annotated_object()` (in module *optimeed.core.myjson*), 44
- `_is_feasible()` (in module *optimeed.optimize.optiAlgorithms.pyswarm*), 74
- `_is_feasible()` (in module *optimeed.optimize.optiAlgorithms.pyswarm.pso*), 73
- `_is_object_selected()` (in module *optimeed.visualize.displayCollections*), 132
- `_isclass()` (in module *optimeed.core*), 65
- `_isclass()` (in module *optimeed.core.myjson*), 43
- `_iter_items()` (*Widget\_listWithSearch* method), 127, 131, 156
- `_iter_items()` (*Widget\_tableWithSearch* method), 129, 131, 158
- `_latex_template` (in module *optimeed.core.ansi2html.converter*), 28
- `_map_index_to_file()` (*Performance\_ListDataStruct* method), 34, 55
- `_needs_extra_newline()` (in module *optimeed.core.ansi2html.converter*), 28
- `_normalize_colors()` (in module *optimeed.visualize.graphs.traceVisual*), 95
- `_object_to_FQCN()` (in module *optimeed.core*), 65
- `_object_to_FQCN()` (in module *optimeed.core.myjson*), 43
- `_pack_options()` (*Option\_class* method), 23, 45, 63



- `_remove_index_from_show()` (*\_AnimationTrace method*), 101
  - `_remove_item_selector()` (*CollectionDisplayer method*), 132, 138
  - `_reset_colors()` (*CollectionDisplayer method*), 133, 138
  - `_save_module_tree()` (*Performance\_ListDataStruct method*), 34, 55
  - `_select_and_apply_action()` (in module *optimeed.visualize.displayCollections*), 132
  - `_selector_to()` (*CollectionDisplayer method*), 132, 138
  - `_set_study()` (*SensitivityDisplayer method*), 134, 139
  - `_set_x()` (*CollectionDisplayer method*), 132, 138
  - `_set_y()` (*CollectionDisplayer method*), 132, 138
  - `_set_z()` (*CollectionDisplayer method*), 132, 138
  - `_updateLabel()` (*myAxis method*), 94
  - `_workspace_path` (in module *optimeed.core*), 49
  - `_workspace_path` (in module *optimeed.core.tools*), 47
- ## A
- `activateMaterialProperties()` (*MaterialRenderingProperties method*), 119, 122, 155
  - `add()` (*SeveralTerminationCondition method*), 78
  - `add_action_on_click()` (in module *optimeed.visualize*), 146
  - `add_action_on_click()` (in module *optimeed.visualize.fastPlot*), 136
  - `add_action_on_click()` (*WindowHolders method*), 135, 145
  - `add_collection()` (*CollectionDisplayer method*), 132, 138
  - `add_collection()` (*LinkDataGraph method*), 41, 60
  - `add_data()` (*GraphVisual method*), 91
  - `add_data()` (*ListDataStruct method*), 24, 33, 54
  - `add_data()` (*Performance\_ListDataStruct method*), 34, 54
  - `add_data()` (*SensitivityResults method*), 21
  - `add_data_to_collection()` (*CollectionExporter-GUI method*), 104
  - `add_element()` (*\_AnimationTrace method*), 101
  - `add_elementToTrace()` (*AnimationGUI method*), 101
  - `add_feature()` (*GraphVisual method*), 91
  - `add_graph()` (*Graphs method*), 39, 59, 144
  - `add_graph()` (*LinkDataGraph method*), 42, 61
  - `add_index_to_show()` (*\_AnimationTrace method*), 101
  - `add_json_data()` (*Performance\_ListDataStruct method*), 34, 55
  - `add_modified_paintElem()` (*TraceVisual.\_ModifiedPaintElem method*), 95
  - `add_opti_project()` (*ViewOptimizationResults method*), 137, 140
  - `add_option()` (*Option\_class method*), 23, 45, 63
  - `add_parameters()` (*OptiHistoric.\_LogParams method*), 82, 89
  - `add_plot()` (*\_PlotHolders method*), 135, 145
  - `add_plot()` (*WindowHolders method*), 135, 145
  - `add_point()` (*ConvergenceManager method*), 75
  - `add_point()` (*Data method*), 38, 57, 143
  - `add_prefix_attribute_name()` (*Optimization-Variable method*), 82
  - `add_study()` (*SensitivityDisplayer method*), 134, 138
  - `add_suffix_to_path()` (in module *optimeed.core*), 50
  - `add_suffix_to_path()` (in module *optimeed.core.tools*), 48
  - `add_terminationCondition()` (*MultiObjective\_GA method*), 79, 80, 87
  - `add_trace()` (*AnimationGUI method*), 101
  - `add_trace()` (*Graph method*), 38, 58
  - `add_trace()` (*Graphs method*), 39, 58, 143
  - `add_trace()` (*GraphVisual method*), 91
  - `add_trace_firstGraph()` (*Graphs method*), 39, 58, 143
  - `add_update_method()` (*Graphs method*), 39, 59, 144
  - `add_visual()` (*OnClick\_representDevice.DataInformationVisuals method*), 109, 115, 148
  - `addItem()` (*myGraphicsLayout method*), 93
  - `addItem()` (*myLegend method*), 94
  - `adjust()` (*\_State method*), 28
  - ALGORITHM (*NLOpt\_Algorithm attribute*), 75
  - AlgorithmInterface (class in *optimeed.optimize.optiAlgorithms.algorithmInterface*), 76
  - `analyse_sobol_convergence()` (in module *optimeed consolidate*), 26
  - `analyse_sobol_convergence()` (in module *optimeed consolidate.sensitivity\_analysis*), 22
  - `analyse_sobol_create_array()` (in module *optimeed consolidate.sensitivity\_analysis*), 22
  - `analyse_sobol_plot_2ndOrder_indices()` (in module *optimeed.visualize*), 139
  - `analyse_sobol_plot_2ndOrder_indices()` (in module *optimeed.visualize.displaySensitivity*), 134
  - `analyse_sobol_plot_convergence()` (in module *optimeed.visualize*), 139
  - `analyse_sobol_plot_convergence()` (in module *optimeed.visualize.displaySensitivity*), 134
  - `analyse_sobol_plot_indices()` (in module *optimeed.visualize*), 139
  - `analyse_sobol_plot_indices()` (in module *optimeed.visualize.displaySensitivity*), 134

- `Animate_lines` (class in *optimeed.visualize*), 153
- `Animate_lines` (class in *optimeed.visualize.onclick*), 116
- `Animate_lines` (class in *optimeed.visualize.onclick.animation\_examples*), 103
- `Animate_lines_and_text` (class in *optimeed.visualize*), 154
- `Animate_lines_and_text` (class in *optimeed.visualize.onclick*), 117
- `Animate_lines_and_text` (class in *optimeed.visualize.onclick.animation\_examples*), 103
- `Animate_openGL` (class in *optimeed.visualize*), 153
- `Animate_openGL` (class in *optimeed.visualize.onclick*), 117
- `Animate_openGL` (class in *optimeed.visualize.onclick.animation\_examples*), 102
- `Animate_openGL_and_text` (class in *optimeed.visualize*), 154
- `Animate_openGL_and_text` (class in *optimeed.visualize.onclick*), 117
- `Animate_openGL_and_text` (class in *optimeed.visualize.onclick.animation\_examples*), 102
- `AnimationGUI` (class in *optimeed.visualize.onclick.animationGUI*), 101
- `Ansi2HTMLConverter` (class in *optimeed.core.ansi2html*), 30
- `Ansi2HTMLConverter` (class in *optimeed.core.ansi2html.converter*), 29
- `ANSI_BACKGROUND_256` (in module *optimeed.core.ansi2html.converter*), 28
- `ANSI_BACKGROUND_CUSTOM_MAX` (in module *optimeed.core.ansi2html.converter*), 28
- `ANSI_BACKGROUND_CUSTOM_MIN` (in module *optimeed.core.ansi2html.converter*), 28
- `ANSI_BACKGROUND_DEFAULT` (in module *optimeed.core.ansi2html.converter*), 28
- `ANSI_BACKGROUND_HIGH_INTENSITY_MAX` (in module *optimeed.core.ansi2html.converter*), 28
- `ANSI_BACKGROUND_HIGH_INTENSITY_MIN` (in module *optimeed.core.ansi2html.converter*), 28
- `ANSI_BLINK_FAST` (in module *optimeed.core.ansi2html.converter*), 27
- `ANSI_BLINK_OFF` (in module *optimeed.core.ansi2html.converter*), 27
- `ANSI_BLINK_SLOW` (in module *optimeed.core.ansi2html.converter*), 27
- `ANSI_CROSSED_OUT_OFF` (in module *optimeed.core.ansi2html.converter*), 27
- `ANSI_CROSSED_OUT_ON` (in module *optimeed.core.ansi2html.converter*), 27
- `ANSI_FOREGROUND_256` (in module *optimeed.core.ansi2html.converter*), 28
- `ANSI_FOREGROUND_CUSTOM_MAX` (in module *optimeed.core.ansi2html.converter*), 28
- `ANSI_FOREGROUND_CUSTOM_MIN` (in module *optimeed.core.ansi2html.converter*), 28
- `ANSI_FOREGROUND_DEFAULT` (in module *optimeed.core.ansi2html.converter*), 28
- `ANSI_FOREGROUND_HIGH_INTENSITY_MAX` (in module *optimeed.core.ansi2html.converter*), 28
- `ANSI_FOREGROUND_HIGH_INTENSITY_MIN` (in module *optimeed.core.ansi2html.converter*), 28
- `ANSI_FULL_RESET` (in module *optimeed.core.ansi2html.converter*), 27
- `ANSI_INTENSITY_INCREASED` (in module *optimeed.core.ansi2html.converter*), 27
- `ANSI_INTENSITY_NORMAL` (in module *optimeed.core.ansi2html.converter*), 27
- `ANSI_INTENSITY_REDUCED` (in module *optimeed.core.ansi2html.converter*), 27
- `ANSI_NEGATIVE_OFF` (in module *optimeed.core.ansi2html.converter*), 28
- `ANSI_NEGATIVE_ON` (in module *optimeed.core.ansi2html.converter*), 28
- `ANSI_STYLE_ITALIC` (in module *optimeed.core.ansi2html.converter*), 27
- `ANSI_STYLE_NORMAL` (in module *optimeed.core.ansi2html.converter*), 27
- `ANSI_UNDERLINE_OFF` (in module *optimeed.core.ansi2html.converter*), 27
- `ANSI_UNDERLINE_ON` (in module *optimeed.core.ansi2html.converter*), 27
- `ANSI_VISIBILITY_OFF` (in module *optimeed.core.ansi2html.converter*), 27
- `ANSI_VISIBILITY_ON` (in module *optimeed.core.ansi2html.converter*), 27
- `app` (in module *optimeed.visualize.process\_mainloop*), 137
- `append()` (*MultiList* method), 72
- `apply_module_tree_to_dict()` (in module *optimeed.core*), 52, 66
- `apply_module_tree_to_dict()` (in module *optimeed.core.myjson*), 44
- `apply_palette()` (*GraphVisual* method), 91
- `apply_regex()` (*Ansi2HTMLConverter* method), 29, 30
- `apply_width_sample()` (*myLegend* method), 94
- `applyEquation()` (in module *optimeed.core*), 49
- `applyEquation()` (in module *optimeed.core.tools*), 47
- `arithmeticEval()` (in module *optimeed.core*), 50
- `arithmeticEval()` (in module *optimeed.core.tools*), 47
- `attributeName` (*OptimizationVariable* attribute), 82

[attrs\(\)](#) (*Ansi2HTMLConverter method*), 29, 30  
[AutosaveStruct](#) (*class in optimeed consolidate*), 24  
[AutosaveStruct](#) (*class in optimeed core*), 53  
[AutosaveStruct](#) (*class in optimeed core collection*), 32  
[axis\\_equal\(\)](#) (*\_PlotHolders method*), 135, 145  
[axis\\_equal\(\)](#) (*GraphVisual method*), 92  
[axis\\_equal\(\)](#) (*in module optimeed visualize*), 146  
[axis\\_equal\(\)](#) (*in module optimeed visualize fastPlot*), 136  
[axis\\_equal\(\)](#) (*WindowHolders method*), 135, 145  
[axisangle\\_to\\_q\(\)](#) (*in module optimeed visualize openGL quaternions*), 121

## B

[Base\\_Option](#) (*class in optimeed core*), 62  
[Base\\_Option](#) (*class in optimeed core options*), 44  
[Binary\\_OptimizationVariable](#) (*class in optimeed optimize*), 88  
[Binary\\_OptimizationVariable](#) (*class in optimeed optimize optiVariable*), 83  
[blackOnly\(\)](#) (*in module optimeed core*), 55  
[blackOnly\(\)](#) (*in module optimeed core color\_palette*), 35  
[BLUE](#) (*text\_format attribute*), 47, 49  
[Blue\\_material](#) (*in module optimeed visualize*), 155  
[Blue\\_material](#) (*in module optimeed visualize openGL*), 122  
[Blue\\_material](#) (*in module optimeed visualize openGL materials*), 119  
[BOLD](#) (*text\_format attribute*), 47, 49  
[boundingRect\(\)](#) (*\_LineItem method*), 108  
[Brass\\_material](#) (*in module optimeed visualize*), 155  
[Brass\\_material](#) (*in module optimeed visualize openGL*), 122  
[Brass\\_material](#) (*in module optimeed visualize openGL materials*), 119  
[Bronze\\_material](#) (*in module optimeed visualize*), 155  
[Bronze\\_material](#) (*in module optimeed visualize openGL*), 122  
[Bronze\\_material](#) (*in module optimeed visualize openGL materials*), 119

## C

[cancel\\_selector\(\)](#) (*Onselect\_highlight method*), 123, 124, 155  
[cancel\\_selector\(\)](#) (*Onselect\_newTrace method*), 123, 125, 156  
[cancel\\_selector\(\)](#) (*Onselect\_splitTrace method*), 124, 125, 156  
[cart2pol\(\)](#) (*in module optimeed core*), 63  
[cart2pol\(\)](#) (*in module optimeed core additional\_tools*), 31

[cellChanged](#) (*Widget\_tableWithSearch attribute*), 129, 131, 157  
[Characterization](#) (*class in optimeed optimize*), 85  
[Characterization](#) (*class in optimeed optimize characterization*), 67  
[Characterization](#) (*class in optimeed optimize characterization characterization*), 67  
[check\\_if\\_must\\_plot\(\)](#) (*in module optimeed visualize displayOptimization*), 133  
[Chrome\\_material](#) (*in module optimeed visualize*), 155  
[Chrome\\_material](#) (*in module optimeed visualize openGL*), 122  
[Chrome\\_material](#) (*in module optimeed visualize openGL materials*), 119  
[CLASS\\_TAG](#) (*in module optimeed core*), 65  
[CLASS\\_TAG](#) (*in module optimeed core myjson*), 43  
[CLIC\\_LEFT](#) (*in module optimeed visualize openGL contextHandler*), 118  
[CLIC\\_RIGHT](#) (*in module optimeed visualize openGL contextHandler*), 118  
[clone\(\)](#) (*ListDataStruct method*), 24, 33, 53  
[clone\(\)](#) (*Performance\_ListDataStruct method*), 34, 54  
[close\(\)](#) (*MyMultiprocessEvaluator method*), 79  
[close\\_windows\(\)](#) (*OptimizationDisplayer method*), 133, 140  
[closeEvent\(\)](#) (*AnimationGUI method*), 102  
[closeEvent\(\)](#) (*MainWindow method*), 136, 141  
[cmapToColormap\(\)](#) (*in module optimeed visualize graphs colormap\_pyqtgraph*), 90  
[CollectionDisplayer](#) (*class in optimeed visualize*), 138  
[CollectionDisplayer](#) (*class in optimeed visualize displayCollections*), 132  
[CollectionExporterGUI](#) (*class in optimeed visualize onclick.collectionExporterGUI*), 103  
[color\(\)](#) (*in module optimeed core ansi2html.style*), 29  
[color\\_component\(\)](#) (*in module optimeed core ansi2html.style*), 29  
[compute\(\)](#) (*\_Objective method*), 19  
[compute\(\)](#) (*Characterization method*), 67, 85  
[compute\(\)](#) (*FastObjCons method*), 69, 70, 86  
[compute\(\)](#) (*HyperVolume method*), 71  
[compute\(\)](#) (*Monobjective\_PSO method*), 76, 81, 87  
[compute\(\)](#) (*MultiObjective\_GA method*), 79, 80, 86  
[compute\(\)](#) (*NLOpt\_Algorithm method*), 75  
[constraints](#) (*OptiHistoric.\_pointData attribute*), 81, 89  
[constraints\\_per\\_step](#) (*EvolutionaryConvergence*

- attribute), 71, 73
- contains\_trace() (*AnimationGUI method*), 102
- ContextHandler (class in *optimeed.visualize.openGL.contextHandler*), 118
- ContourPlot (class in *optimeed.core*), 60
- ContourPlot (class in *optimeed.core.graphs3*), 40
- ContourPlot (class in *optimeed.visualize*), 146
- ConvergenceManager (class in *optimeed.optimize.optiAlgorithms.NLOpt\_Algorithm*), 75
- ConvergenceTerminationCondition (class in *optimeed.optimize.optiAlgorithms.multiObjective\_GA*), 78
- convert() (*Ansi2HTMLConverter method*), 29, 30
- convert\_color() (in module *optimeed.core*), 64
- convert\_color() (in module *optimeed.core.additional\_tools*), 32
- convert\_color\_with\_alpha() (in module *optimeed.core*), 55, 64
- convert\_color\_with\_alpha() (in module *optimeed.core.additional\_tools*), 32
- convert\_linestyle() (in module *optimeed.core.tikzTranslator*), 46
- convert\_marker() (in module *optimeed.core.tikzTranslator*), 46
- convert\_to\_gridplot() (in module *optimeed.core*), 60
- convert\_to\_gridplot() (in module *optimeed.core.graphs3*), 40
- Copper\_material (in module *optimeed.visualize*), 155
- Copper\_material (in module *optimeed.visualize.openGL*), 122
- Copper\_material (in module *optimeed.visualize.openGL.materials*), 119
- create\_main\_window() (*OptimizationDisplayer method*), 133, 140
- create\_unique\_dirname() (in module *optimeed.core*), 49
- create\_unique\_dirname() (in module *optimeed.core.tools*), 47
- createWidget() (*Plugin\_listWithSearch method*), 127
- createWidget() (*Plugin\_tableWithSearch method*), 129
- curr\_index() (*OnClick\_representDevice.DataInformationVisuals method*), 109, 115, 148
- CursorMoveUp (class in *optimeed.core.ansi2html.converter*), 28
- CYAN (text\_format attribute), 47, 49
- Cyan\_material (in module *optimeed.visualize*), 155
- Cyan\_material (in module *optimeed.visualize.openGL*), 122
- Cyan\_material (in module *optimeed.visualize.openGL.materials*), 120
- ## D
- dark2() (in module *optimeed.core*), 55
- dark2() (in module *optimeed.core.color\_palette*), 35
- DARKCYAN (text\_format attribute), 47, 49
- Data (class in *optimeed.core*), 55
- Data (class in *optimeed.core.graphs*), 36
- Data (class in *optimeed.visualize*), 141
- DataStruct\_Interface (class in *optimeed.core*), 53
- DataStruct\_Interface (class in *optimeed.core.collection*), 32
- decode\_str\_json() (in module *optimeed.core*), 52, 66
- decode\_str\_json() (in module *optimeed.core.myjson*), 44
- deep\_sizeof() (in module *optimeed.core*), 51
- deep\_sizeof() (in module *optimeed.core.tools*), 49
- default (in module *optimeed.optimize.optimizer*), 84
- default\_colormap (in module *optimeed.visualize.graphs.traceVisual*), 95
- default\_palette() (in module *optimeed.core*), 55
- default\_palette() (in module *optimeed.core.color\_palette*), 35
- delete() (*GraphVisual method*), 92
- delete() (*Widget\_graphsVisualLite method*), 98, 100, 147
- delete\_all() (*\_AnimationTrace method*), 101
- delete\_all() (*AnimationGUI method*), 101
- delete\_clicked\_item() (*LinkDataGraph method*), 43, 62
- delete\_clicked\_items() (*LinkDataGraph method*), 43, 62
- delete\_graph() (*Widget\_graphsVisualLite method*), 98, 100, 147
- delete\_indices\_from\_list() (in module *optimeed.core*), 51, 52
- delete\_indices\_from\_list() (in module *optimeed.core.tools*), 48
- delete\_key\_widgets() (*Animate\_lines method*), 103, 116, 153
- delete\_key\_widgets() (*Animate\_openGL method*), 102, 117, 153
- delete\_lines() (*Widget\_lineDrawer method*), 126, 130, 157
- delete\_point() (*AnimationGUI method*), 101
- delete\_point() (*Data method*), 38, 57, 143
- delete\_points\_at\_indices() (*ListDataStruct method*), 24, 33, 54
- delete\_points\_at\_indices() (*Performance\_ListDataStruct method*), 35, 55



`delete_trace()` (*GraphVisual* method), 92  
`delete_visual()` (*OnClick\_representDevice.DataInformationVisualize* method), 109, 115, 148  
`derivate()` (in module *optimeed.core*), 63  
`derivate()` (in module *optimeed.core.additional\_tools*), 31  
`DeviceDrawerInterface` (class in *optimeed.visualize*), 154  
`DeviceDrawerInterface` (class in *optimeed.visualize.openGL*), 121  
`DeviceDrawerInterface` (class in *optimeed.visualize.openGL.deviceDrawerInterface*), 119  
`disableLogs()` (in module *optimeed.core*), 51  
`disableLogs()` (in module *optimeed.core.commonImport*), 35  
`display_distance()` (*OnClick\_measure* method), 109, 114, 145, 151  
`display_graphs()` (*OptimizationDisplayer* method), 133, 140  
`display_graphs()` (*ViewOptimizationResults* method), 137, 140  
`dist()` (in module *optimeed.core*), 64  
`dist()` (in module *optimeed.core.additional\_tools*), 31  
`DIVISION_OUTER` (*MultiObjective\_GA* attribute), 79, 80, 86  
`do_fit()` (in module *optimeed consolidate*), 26  
`do_fit()` (in module *optimeed consolidate.fit*), 20  
`do_generate_figure()` (in module *optimeed.core.tikzTranslator*), 46  
`do_MathsToPhys()` (*Binary\_OptimizationVariable* method), 83, 88  
`do_MathsToPhys()` (*Integer\_OptimizationVariable* method), 83, 88  
`do_MathsToPhys()` (*OptimizationVariable* method), 82  
`do_MathsToPhys()` (*Real\_OptimizationVariable* method), 83, 88  
`do_preamble()` (in module *optimeed.core.tikzTranslator*), 46  
`do_preamble3D()` (in module *optimeed.core.tikzTranslator*), 46  
`do_specific_axis_options()` (in module *optimeed.core.tikzTranslator*), 46  
`do_specific_trace_options()` (in module *optimeed.core.tikzTranslator*), 46  
`doubleValueChanged` (*widget\_doubleSlider* attribute), 126  
`draw_2Dring()` (in module *optimeed.visualize.openGL.openGL\_library*), 120  
`draw_2Dring_diff_angle()` (in module *optimeed.visualize.openGL.openGL\_library*), 120  
`draw_carved_disk()` (in module *optimeed.visualize.openGL.openGL\_library*), 120  
`draw_closedPolygon()` (in module *optimeed.visualize.openGL.openGL\_library*), 120  
`draw_cylinder()` (in module *optimeed.visualize.openGL.openGL\_library*), 120  
`draw_disk()` (in module *optimeed.visualize.openGL.openGL\_library*), 120  
`draw_extrudeZ()` (in module *optimeed.visualize.openGL.openGL\_library*), 120  
`draw_lines()` (in module *optimeed.visualize.openGL.openGL\_library*), 120  
`draw_part_cylinder()` (in module *optimeed.visualize.openGL.openGL\_library*), 120  
`draw_part_cylinder_throat()` (in module *optimeed.visualize.openGL.openGL\_library*), 120  
`draw_part_disk()` (in module *optimeed.visualize.openGL.openGL\_library*), 120  
`draw_part_disk_diff_angles()` (in module *optimeed.visualize.openGL.openGL\_library*), 120  
`draw_rectangle()` (in module *optimeed.visualize.openGL.openGL\_library*), 120  
`draw_simple_rectangle()` (in module *optimeed.visualize.openGL.openGL\_library*), 120  
`draw_spiral()` (in module *optimeed.visualize.openGL.openGL\_library*), 120  
`draw_spiralFront()` (in module *optimeed.visualize.openGL.openGL\_library*), 120  
`draw_spiralFull()` (in module *optimeed.visualize.openGL.openGL\_library*), 120  
`draw_spiralSheet()` (in module *optimeed.visualize.openGL.openGL\_library*), 120  
`draw_triList()` (in module *optimeed.visualize.openGL.openGL\_library*), 120  
`draw_tubeSheet()` (in module *optimeed.visualize.openGL.openGL\_library*), 120  
`drawWireTube()` (in module *optimeed.visualize.openGL.openGL\_library*), 120

## E

Emerald\_material (in module *optimeed.visualize*), 155  
 Emerald\_material (in module *optimeed.visualize.openGL*), 122  
 Emerald\_material (in module *optimeed.visualize.openGL.materials*), 119  
 emitDoubleValueChanged() (wid-  
 get\_doubleSlider method), 126  
 enableLogs() (in module *optimeed.core*), 51  
 enableLogs() (in module *optimeed.core.commonImport*), 35  
 encode\_str\_json() (in module *optimeed.core*), 52, 66  
 encode\_str\_json() (in module *optimeed.core.myjson*), 44  
 END (text\_format attribute), 47, 49  
 evaluate() (*\_Evaluator* method), 84  
 evaluate() (in module *optimeed consolidate.sensitivity\_analysis\_evaluation*), 23  
 evaluate() (*MyProblem* method), 77  
 evaluate() (*Parametric\_analysis* method), 21, 25  
 evaluate\_all() (*MyMapEvaluator* method), 73, 74, 78  
 evaluate\_all() (*MyMultiprocessEvaluator* method), 74, 79  
 evaluate\_sensitivities() (in module *optimeed consolidate*), 26  
 evaluate\_sensitivities() (in module *optimeed consolidate.sensitivity\_analysis*), 22  
 eventFilter() (*Widget\_image* method), 126, 130, 156  
 EvolutionaryConvergence (class in *optimeed.optimize.optiAlgorithms.convergence*), 73  
 EvolutionaryConvergence (class in *optimeed.optimize.optiAlgorithms.convergence.evolutionaryConvergence*), 71  
 EXCLUDED\_TAGS (in module *optimeed.core*), 65  
 EXCLUDED\_TAGS (in module *optimeed.core.myjson*), 43  
 export\_picture() (*AnimationGUI* method), 102  
 export\_str() (*Data* method), 38, 58, 143  
 export\_str() (*Graph* method), 38, 58  
 export\_str() (*Graphs* method), 39, 59, 144  
 export\_svg() (*Widget\_graphsVisualLite* method), 98, 99, 147  
 export\_tikz() (*Widget\_graphsVisualLite* method), 98, 100, 147  
 export\_to\_tikz\_contour\_plot() (in module *optimeed.core*), 66  
 export\_to\_tikz\_contour\_plot() (in module *optimeed.core.tikzTranslator*), 46

export\_to\_tikz\_groupGraphs() (in module *optimeed.core*), 66  
 export\_to\_tikz\_groupGraphs() (in module *optimeed.core.tikzTranslator*), 46  
 export\_txt() (*Widget\_graphsVisualLite* method), 98, 99, 147  
 export\_widget() (*Animate\_lines* method), 103, 116, 153  
 export\_widget() (*Animate\_openGL* method), 102, 117, 153  
 exportCollection() (*CollectionExporterGUI* method), 104  
 exportGraphs() (*Widget\_graphsVisualLite* method), 98, 99, 147  
 extend() (*MultiList* method), 72  
 extract\_collection\_from\_indices() (*ListDataStruct* method), 24, 33, 53  
 extract\_collection\_from\_indices() (*Performance\_ListDataStruct* method), 34, 54

## F

fast\_LUT\_interpolation (class in *optimeed.core*), 63  
 fast\_LUT\_interpolation (class in *optimeed.core.additional\_tools*), 30  
 fast\_update() (*GraphVisual* method), 92  
 fast\_update() (*Widget\_graphsVisualLite* method), 98, 99, 147  
 FastObjCons (class in *optimeed.optimize*), 86  
 FastObjCons (class in *optimeed.optimize.objAndCons*), 70  
 FastObjCons (class in *optimeed.optimize.objAndCons.fastObjCons*), 69  
 figure() (in module *optimeed.visualize*), 146  
 figure() (in module *optimeed.visualize.fastPlot*), 135  
 FilledContourPlot (class in *optimeed.core*), 60  
 FilledContourPlot (class in *optimeed.core.graphs3*), 40  
 FilledContourPlot (class in *optimeed.visualize*), 146  
 find\_all\_colors() (in module *optimeed.core.tikzTranslator*), 46  
 find\_and\_replace() (in module *optimeed.core*), 49  
 find\_and\_replace() (in module *optimeed.core.tools*), 47  
 force\_hide\_row() (*Widget\_tableWithSearch* method), 129, 131, 157  
 format\_escape\_char() (in module *optimeed.core.tikzTranslator*), 46  
 format\_Griddata() (in module *optimeed.core.tikzTranslator*), 46  
 format\_scatterdata() (in module *optimeed.core.tikzTranslator*), 46

frame\_selector() (*AnimationGUI method*), 101  
 fromMathsToPhys() (*MathsToPhysics method*), 68, 69, 85  
 fromPhysToMaths() (*MathsToPhysics method*), 68, 69, 85

## G

gather\_embarrassingly\_parallel\_sensitivity() (in module *optimeed consolidate*), 27  
 gather\_embarrassingly\_parallel\_sensitivity() (in module *optimeed consolidate.sensitivity\_analysis*), 22  
 generate() (*MyGenerator method*), 77  
 generate\_optimizationGraphs() (*OptimizationDisplayer method*), 133, 139  
 get() (*\_AnimationTrace.AnimationElement method*), 101  
 get\_2D\_pareto() (in module *optimeed.core*), 50  
 get\_2D\_pareto() (in module *optimeed.core.tools*), 48  
 get\_additional\_attributes\_to\_save() (*OptimizerSettings method*), 84, 89  
 get\_additional\_attributes\_to\_save() (*SaveableObject method*), 43, 65  
 get\_additional\_attributes\_to\_save() (*SensitivityParameters method*), 22, 26  
 get\_additional\_attributes\_to\_save() (*SensitivityResults method*), 21  
 get\_additional\_attributes\_to\_save\_list() (*OptimizerSettings method*), 84, 89  
 get\_additional\_attributes\_to\_save\_list() (*SaveableObject method*), 43, 65  
 get\_all\_figures() (in module *optimeed.visualize*), 146  
 get\_all\_figures() (in module *optimeed.visualize.fastPlot*), 136  
 get\_all\_figures() (*WindowHolders method*), 135, 145  
 get\_all\_graphs() (*Graphs method*), 39, 59, 144  
 get\_all\_graphs\_ids() (*Graphs method*), 39, 59, 144  
 get\_all\_graphsVisual() (*WindowHolders method*), 98, 100, 147  
 get\_all\_traces() (*Graph method*), 38, 58  
 get\_all\_traces() (*GraphVisual method*), 92  
 get\_all\_traces\_ids() (*Graph method*), 38, 58  
 get\_alpha() (*Data method*), 36, 56, 142  
 get\_analyzed\_attribute() (*ParameterParameter method*), 20, 25  
 get\_attribute\_name() (*OptimizationVariable method*), 82  
 get\_axis() (*GraphVisual method*), 91  
 get\_base\_pen() (*\_AnimationTrace method*), 101  
 get\_base\_pen() (*TraceVisual method*), 95  
 get\_base\_symbol() (*TraceVisual method*), 96  
 get\_base\_symbol\_brush() (*TraceVisual method*), 95  
 get\_base\_symbol\_pen() (*TraceVisual method*), 95  
 get\_best\_devices\_without\_reevaluating() (*OptiHistoric method*), 82, 89  
 get\_brushes() (*TraceVisual method*), 96  
 get\_charac() (*OptimizerSettings method*), 84, 89  
 get\_charac() (*SensitivityParameters method*), 22, 26  
 get\_choices() (*Base\_Option method*), 44, 62  
 get\_choices() (*Option\_bool method*), 45, 62  
 get\_clicked\_item() (*LinkDataGraph method*), 42, 61  
 get\_clicked\_items() (*LinkDataGraph method*), 42, 62  
 get\_collection() (*LinkDataGraph method*), 42, 61  
 get\_collection\_from\_graph() (*LinkDataGraph method*), 42, 61  
 get\_color() (*Data method*), 36, 56, 142  
 get\_color() (*ScatterPlot3 method*), 40, 60, 146  
 get\_color() (*TraceVisual method*), 95  
 get\_color\_alpha() (*Data method*), 36, 56, 142  
 get\_colour\_background() (*DeviceDrawerInterface method*), 119, 121, 154  
 get\_colour\_scalebar() (*DeviceDrawerInterface method*), 119, 121, 154  
 get\_constraints() (*OptimizerSettings method*), 84, 89  
 get\_convergence() (*\_OptiProjectLoader method*), 137  
 get\_convergence() (*Monobjective\_PSO method*), 77, 81, 87  
 get\_convergence() (*MultiObjective\_GA method*), 79, 80, 87  
 get\_convergence() (*NLOpt\_Algorithm method*), 76  
 get\_convergence() (*OptiHistoric method*), 82, 90  
 get\_curr\_plotHolder() (*WindowHolders method*), 135, 145  
 get\_data() (*ListDataStruct method*), 24, 33, 54  
 get\_data() (*TraceVisual method*), 96  
 get\_data\_at\_index() (*ListDataStruct method*), 24, 33, 54  
 get\_data\_at\_index() (*Performance\_ListDataStruct method*), 34, 55  
 get\_data\_generator() (*ListDataStruct method*), 24, 33, 54  
 get\_data\_generator() (*Performance\_ListDataStruct method*), 34, 55  
 get\_data\_link() (*ViewOptimizationResults method*), 137, 140  
 get\_dataIndex\_from\_graphIndex() (*Data method*), 37, 57, 142  
 get\_dataIndices\_from\_graphIndices()

(Data method), 37, 57, 142

get\_datalink() (CollectionDisplayer method), 132, 138

get\_datastruct() (AutosaveStruct method), 25, 33, 53

get\_device() (OptimizerSettings method), 84, 89

get\_device() (SensitivityParameters method), 21, 26

get\_devices() (\_OptiProjectLoader method), 137

get\_devices() (OptiHistoric method), 82, 90

get\_element\_animations() (\_AnimationTrace method), 101

get\_ellipse\_axes() (in module optimeed.core), 64

get\_ellipse\_axes() (in module optimeed.core.additional\_tools), 31

get\_entries\_selected() (Widget\_tableWithSearch method), 129, 131, 158

get\_extrema\_lines() (Widget\_lineDrawer method), 127, 130, 157

get\_filename() (AutosaveStruct method), 24, 32, 53

get\_first\_graph() (Graphs method), 39, 59, 144

get\_graph() (Graphs method), 39, 59, 144

get\_graph() (Widget\_graphsVisualLite method), 98, 100, 147

get\_graph\_and\_trace\_from\_idCollection() (LinkDataGraph method), 43, 62

get\_graphIndex\_from\_dataIndex() (Data method), 37, 57, 142

get\_graphIndices\_from\_dataIndices() (Data method), 37, 57, 142

get\_graphs() (EvolutionaryConvergence method), 71, 73

get\_graphs() (LinkDataGraph method), 42, 61

get\_howToPlotGraph() (LinkDataGraph method), 42, 61

get\_hypervolume() (EvolutionaryConvergence method), 71, 73

get\_hypervolume\_convergence() (EvolutionaryConvergence method), 71, 73

get\_idcollection\_from\_collection() (LinkDataGraph method), 43, 62

get\_idCollection\_from\_graph() (LinkDataGraph method), 42, 61

get\_idCollections() (LinkDataGraph method), 42, 61

get\_idGraphs() (LinkDataGraph method), 42, 61

get\_idPoints\_from\_indices\_in\_collection() (LinkDataGraph method), 43, 62

get\_idTraces() (LinkDataGraph method), 42, 61

get\_index\_selected() (Widget\_listWithSearch method), 127, 130, 156

get\_indices\_points\_to\_plot() (Data method), 38, 58, 143

get\_indices\_to\_show() (\_AnimationTrace method), 101

get\_inkscape\_version() (in module optimeed.core), 66

get\_inkscape\_version() (in module optimeed.core.inkscape\_manager), 41

get\_interesting\_elements() (Animate\_lines method), 103, 116, 153

get\_interesting\_elements() (Animate\_openGL\_and\_text method), 102, 117, 154

get\_invert\_permutations() (Data method), 37, 57, 142

get\_item() (Widget\_tableWithSearch method), 129, 131, 158

get\_json\_module\_tree\_from\_dict() (in module optimeed.core), 52, 65

get\_json\_module\_tree\_from\_dict() (in module optimeed.core.myjson), 44

get\_kwargs() (\_OptiProjectLoader method), 137

get\_label() (Plot3D\_Generic method), 40, 60

get\_label\_pos() (myAxis method), 94

get\_layout\_buttons() (Widget\_graphsVisualLite method), 98, 100, 147

get\_legend() (Data method), 37, 57, 143

get\_legend() (GraphVisual method), 91

get\_legend() (Plot3D\_Generic method), 40, 60

get\_length() (ListDataStruct method), 24, 33, 53

get\_length() (TraceVisual method), 96

get\_length\_data() (Data method), 36, 56, 141

get\_levels() (ContourPlot method), 40, 60, 146

get\_lim() (Plot3D\_Generic method), 40, 60

get\_linestyle() (Data method), 38, 58, 143

get\_list\_attributes() (ListDataStruct\_Interface method), 32, 53

get\_logopti() (\_OptiProjectLoader method), 137

get\_logopti() (OptiHistoric method), 82, 90

get\_M2P() (OptimizerSettings method), 84, 89

get\_M2P() (SensitivityParameters method), 21, 26

get\_max\_value() (Integer\_OptimizationVariable method), 83, 88

get\_max\_value() (Real\_OptimizationVariable method), 83, 88

get\_meta() (Data method), 36, 56, 142

get\_min\_value() (Integer\_OptimizationVariable method), 83, 88

get\_min\_value() (Real\_OptimizationVariable method), 83, 88

get\_nadir\_point() (EvolutionaryConvergence method), 71, 73

get\_name() (Base\_Option method), 44, 62

get\_name() (FastObjCons method), 69, 70, 86

get\_name() (InterfaceObjCons method), 70, 86

get\_name() (OnClick\_animate method), 104, 111, 148

get\_name() (OnClick\_changeSymbol method), 105,



112, 149  
 get\_name() (*OnClick\_copySomething method*), 105, 112, 149  
 get\_name() (*OnClick\_delete method*), 106, 112, 149  
 get\_name() (*OnClick\_exportCollection method*), 106, 113, 150  
 get\_name() (*OnClick\_exportToTxt method*), 107, 113, 150  
 get\_name() (*OnClick\_exportTrace method*), 107, 113, 150  
 get\_name() (*OnClick\_extractPareto method*), 108, 114, 151  
 get\_name() (*OnClick\_measure method*), 109, 114, 145, 151  
 get\_name() (*OnClick\_removeTrace method*), 109, 114, 151  
 get\_name() (*OnClick\_representDevice method*), 110, 115, 148  
 get\_name() (*OnClick\_tojson method*), 110, 115, 152  
 get\_name() (*Onselect\_highlight method*), 123, 124, 155  
 get\_name() (*Onselect\_newTrace method*), 123, 125, 156  
 get\_name() (*Onselect\_splitTrace method*), 124, 125, 156  
 get\_name\_selected() (*Widget\_listWithSearch method*), 127, 130, 156  
 get\_nb\_objectives() (*EvolutionaryConvergence method*), 71, 73  
 get\_nbr\_elements() (*ListDataStruct method*), 24, 33, 54  
 get\_nbr\_elements() (*Performance\_ListDataStruct method*), 34, 55  
 get\_nbr\_objectives() (*\_OptiProjectLoader method*), 137  
 get\_ND\_pareto() (*in module optimeed.core*), 50  
 get\_ND\_pareto() (*in module optimeed.core.tools*), 48  
 get\_new\_index() (*OnClick\_representDevice.DataInformationVisuals method*), 109, 115, 148  
 get\_number\_of\_contours() (*ContourPlot method*), 40, 60, 146  
 get\_number\_of\_elements() (*\_AnimationTrace method*), 101  
 get\_number\_of\_points() (*Data method*), 36, 56, 142  
 get\_object\_attrs() (*in module optimeed.core*), 50  
 get\_object\_attrs() (*in module optimeed.core.tools*), 48  
 get\_objectives() (*OptimizerSettings method*), 84, 89  
 get\_opengl\_options() (*DeviceDrawerInterface method*), 119, 121, 154  
 get\_optimalgorithm() (*OptimizerSettings method*), 84, 89  
 get\_option\_name() (*Option\_class method*), 23, 45, 63  
 get\_option\_value() (*Option\_class method*), 23, 45, 63  
 get\_optivariabes() (*OptimizerSettings method*), 84, 89  
 get\_optivariabes() (*SensitivityParameters method*), 22, 26  
 get\_paramvalues() (*SensitivityParameters method*), 22, 26  
 get\_pareto\_at\_step() (*EvolutionaryConvergence method*), 71, 73  
 get\_pareto\_convergence() (*EvolutionaryConvergence method*), 71, 73  
 get\_path\_to\_inkscape() (*in module optimeed.core*), 66  
 get\_path\_to\_inkscape() (*in module optimeed.core.inkscape\_manager*), 41  
 get\_permutations() (*Data method*), 37, 57, 142  
 get\_PhysToMaths() (*Binary\_OptimizationVariable method*), 83, 88  
 get\_PhysToMaths() (*Integer\_OptimizationVariable method*), 83, 88  
 get\_PhysToMaths() (*OptimizationVariable method*), 82  
 get\_PhysToMaths() (*Real\_OptimizationVariable method*), 83, 88  
 get\_plot\_data() (*Data method*), 37, 56, 142  
 get\_plot\_data() (*GridPlot\_Generic method*), 40, 60  
 get\_plot\_data() (*ScatterPlot3 method*), 40, 60, 146  
 get\_plot\_meta() (*Data method*), 37, 56, 142  
 get\_point() (*TraceVisual method*), 97  
 get\_recursive\_attrs() (*in module optimeed.core*), 50  
 get\_recursive\_attrs() (*in module optimeed.core.tools*), 48  
 get\_sensitivity\_device() (*Parametric\_parameter method*), 20, 25  
 get\_rows\_indices() (*OptiHistoric\_LogParams method*), 82, 89  
 get\_scalar\_convergence\_evolution() (*EvolutionaryConvergence method*), 71, 73  
 get\_sensitivity\_problem() (*in module optimeed consolidate*), 26  
 get\_sensitivity\_problem() (*in module optimeed consolidate.sensitivity\_analysis*), 22  
 get\_shown\_entries() (*Widget\_tableWithSearch method*), 129, 131, 158  
 get\_styles() (*in module optimeed.core.ansi2html.style*), 30  
 get\_symbol() (*Data method*), 37, 57, 143  
 get\_symbol() (*TraceVisual method*), 96

`get_symbolOutline()` (*Data method*), 36, 56, 141  
`get_symbolPens()` (*TraceVisual method*), 97  
`get_symbolsSize()` (*Data method*), 36, 56, 141  
`get_text_to_write()` (*ContextHandler method*), 118  
`get_total_nbr_elements()` (*Performance\_ListDataStruct method*), 34, 54  
`get_trace()` (*Graph method*), 38, 58  
`get_trace()` (*GraphVisual method*), 92  
`get_trace()` (*Widget\_graphsVisualLite method*), 98, 100, 147  
`get_type_class()` (*in module optimeed.core*), 65  
`get_type_class()` (*in module optimeed.core.myjson*), 43  
`get_value()` (*Base\_Option method*), 44, 62  
`get_values()` (*Parametric\_minmax method*), 20, 25  
`get_wgGraphs()` (*\_PlotHolders method*), 135, 145  
`get_wgGraphs()` (*in module optimeed.visualize*), 146  
`get_wgGraphs()` (*in module optimeed.visualize.fastPlot*), 136  
`get_wgGraphs()` (*WindowHolders method*), 135, 145  
`get_widget()` (*Represent\_brut\_attributes method*), 111, 116, 152  
`get_widget()` (*Represent\_image method*), 111, 115, 152  
`get_widget()` (*Represent\_lines method*), 110, 116, 152  
`get_widget()` (*Represent\_opengl method*), 111, 115, 152  
`get_widget()` (*Represent\_txt\_function method*), 110, 116, 153  
`get_width()` (*Data method*), 36, 56, 142  
`get_x()` (*Data method*), 36, 56, 141  
`get_x_label()` (*Data method*), 37, 57, 143  
`get_xlim()` (*Data method*), 36, 56, 141  
`get_y()` (*Data method*), 36, 56, 142  
`get_y_label()` (*Data method*), 37, 57, 143  
`get_ylim()` (*Data method*), 36, 56, 141  
`getAmb3()` (*MaterialRenderingProperties method*), 119, 122, 154  
`getCurrentShow()` (*in module optimeed.core*), 51  
`getCurrentShow()` (*in module optimeed.core.commonImport*), 35  
`getDif3()` (*MaterialRenderingProperties method*), 119, 122, 154  
`GetEar()` (*in module optimeed.visualize.openGL.triangulate\_polygon*), 121  
`getExecPath()` (*in module optimeed.core*), 65  
`getExecPath()` (*in module optimeed.core.myjson*), 43  
`getLength()` (*MultiList method*), 72  
`getLineInfo()` (*in module optimeed.core*), 50  
`getLineInfo()` (*in module optimeed.core.tools*), 48  
`getPath_workspace()` (*in module optimeed.consolidate*), 25  
`getPath_workspace()` (*in module optimeed.core*), 50, 51  
`getPath_workspace()` (*in module optimeed.core.tools*), 48  
`getShin()` (*MaterialRenderingProperties method*), 119, 122, 155  
`getSpec3()` (*MaterialRenderingProperties method*), 119, 122, 154  
`getTxt()` (*Represent\_txt\_function method*), 110, 116, 153  
`Graph` (*class in optimeed.core*), 58  
`Graph` (*class in optimeed.core.graphs*), 38  
`graph_clicked()` (*OnClick\_animate method*), 104, 111, 148  
`graph_clicked()` (*OnClick\_changeSymbol method*), 105, 112, 149  
`graph_clicked()` (*OnClick\_copySomething method*), 105, 112, 149  
`graph_clicked()` (*OnClick\_delete method*), 106, 112, 149  
`graph_clicked()` (*OnClick\_exportCollection method*), 106, 113, 149  
`graph_clicked()` (*OnClick\_exportToTxt method*), 107, 113, 150  
`graph_clicked()` (*OnClick\_exportTrace method*), 107, 113, 150  
`graph_clicked()` (*OnClick\_extractPareto method*), 108, 114, 150  
`graph_clicked()` (*OnClick\_measure method*), 108, 114, 145, 151  
`graph_clicked()` (*OnClick\_removeTrace method*), 109, 114, 151  
`graph_clicked()` (*OnClick\_representDevice method*), 109, 115, 148  
`graph_clicked()` (*OnClick\_tojson method*), 110, 115, 151  
`Graphs` (*class in optimeed.core*), 58  
`Graphs` (*class in optimeed.core.graphs*), 38  
`Graphs` (*class in optimeed.visualize*), 143  
`GraphVisual` (*class in optimeed.visualize.graphs.graphVisual*), 90  
`GREEN` (*text\_format attribute*), 47, 49  
`Green_material` (*in module optimeed.visualize*), 155  
`Green_material` (*in module optimeed.visualize.openGL*), 122  
`Green_material` (*in module optimeed.visualize.openGL.materials*), 120  
`grid_off()` (*GraphVisual method*), 92  
`griddata_found` (*in module optimeed.core*), 59  
`griddata_found` (*in module optimeed.core.graphs3*), 40  
`GridPlot_Generic` (*class in optimeed.core*), 60

GridPlot\_Generic (class in *optimeed.core.graphs3*), 40  
 group() (*Plugin\_listWithSearch* method), 127  
 group() (*Plugin\_tableWithSearch* method), 129

## H

has\_matplotlib (in module *optimeed.visualize.graphs.colormap\_pyqtgraph*), 90  
 has\_scipy (in module *optimeed.core*), 63  
 has\_scipy (in module *optimeed.core.additional\_tools*), 30  
 hasPlotly (in module *optimeed.visualize*), 146  
 hasPlotly (in module *optimeed.visualize.fastPlot3*), 136  
 hide() (*TraceVisual* method), 96  
 hide\_axes() (*GraphVisual* method), 91  
 hide\_points() (*TraceVisual* method), 95  
 hideRow() (*Widget\_tableWithSearch* method), 129, 131, 157  
 HowToPlotGraph (class in *optimeed.core*), 60  
 HowToPlotGraph (class in *optimeed.core.linkDataGraph*), 41  
 hvRecursive() (*HyperVolume* method), 71  
 HyperVolume (class in *optimeed.optimize.optiAlgorithms.convergence.hypervolume*), 71  
 hypervolume\_per\_step (*EvolutionaryConvergence* attribute), 71, 73

## I

icon() (*Plugin\_listWithSearch* method), 127  
 icon() (*Plugin\_tableWithSearch* method), 129  
 includeFile() (*Plugin\_listWithSearch* method), 127  
 includeFile() (*Plugin\_tableWithSearch* method), 129  
 indentParagraph() (in module *optimeed.core*), 50, 52  
 indentParagraph() (in module *optimeed.core.tools*), 48  
 index (*SensitivityResults* attribute), 21  
 index() (in module *optimeed.core.ansi2html.style*), 29  
 index2() (in module *optimeed.core.ansi2html.style*), 29  
 initialize() (*ContextHandler* method), 118  
 initialize() (*ConvergenceTerminationCondition* method), 78  
 initialize() (*MaxTimeTerminationCondition* method), 77  
 initialize() (*Monobjective\_PSO* method), 76, 81, 87  
 initialize() (*MultiObjective\_GA* method), 79, 80, 86  
 initialize() (*NLOpt\_Algorithm* method), 75

initialize() (*Plugin\_listWithSearch* method), 127  
 initialize() (*Plugin\_tableWithSearch* method), 129  
 initialize() (*SeveralTerminationCondition* method), 78  
 initialize\_output\_collection() (*Parametric\_analysis* method), 21, 25  
 initializeGL() (*Widget\_openGL* method), 128, 131, 157  
 inkscape\_svg\_to\_pdf() (in module *optimeed.core*), 66  
 inkscape\_svg\_to\_pdf() (in module *optimeed.core.inkscape\_manager*), 41  
 inkscape\_svg\_to\_png() (in module *optimeed.core*), 66  
 inkscape\_svg\_to\_png() (in module *optimeed.core.inkscape\_manager*), 41  
 inkscape\_version (in module *optimeed.core*), 66  
 inkscape\_version (in module *optimeed.core.inkscape\_manager*), 41  
 Integer\_OptimizationVariable (class in *optimeed.optimize*), 88  
 Integer\_OptimizationVariable (class in *optimeed.optimize.optiVariable*), 83  
 integrate() (in module *optimeed.core*), 64  
 integrate() (in module *optimeed.core.additional\_tools*), 31  
 intensify() (in module *optimeed.core.ansi2html.style*), 29  
 InterfaceCharacterization (class in *optimeed.optimize*), 85  
 InterfaceCharacterization (class in *optimeed.optimize.characterization*), 67  
 InterfaceCharacterization (class in *optimeed.optimize.characterization.interfaceCharacterization*), 67  
 InterfaceConvergence (class in *optimeed.optimize.optiAlgorithms.convergence*), 73  
 InterfaceConvergence (class in *optimeed.optimize.optiAlgorithms.convergence.interfaceConvergence*), 72  
 InterfaceMathsToPhysics (class in *optimeed.optimize*), 86  
 InterfaceMathsToPhysics (class in *optimeed.optimize.mathsToPhysics*), 69  
 InterfaceMathsToPhysics (class in *optimeed.optimize.mathsToPhysics.interfaceMathsToPhysics*), 68  
 InterfaceObjCons (class in *optimeed.optimize*), 86  
 InterfaceObjCons (class in *optimeed.optimize.objAndCons*), 70  
 InterfaceObjCons (class in *optimeed.optimize.objAndCons.interfaceObjCons*), 70

- [interpolate\(\)](#) (*fast\_LUT\_interpolation method*), [31](#), [63](#)  
[interpolate\\_table\(\)](#) (*in module optimeed.core*), [63](#)  
[interpolate\\_table\(\)](#) (*in module optimeed.core.additional\_tools*), [31](#)  
[InTriangle\(\)](#) (*in module optimeed.visualize.openGL.triangulate\_polygon*), [121](#)  
[is\\_empty\(\)](#) (*AnimationGUI method*), [102](#)  
[is\\_empty\(\)](#) (*Graphs method*), [40](#), [59](#), [144](#)  
[IsClockwise\(\)](#) (*in module optimeed.visualize.openGL.triangulate\_polygon*), [121](#)  
[isContained\(\)](#) (*Plugin\_listWithSearch method*), [127](#)  
[isContained\(\)](#) (*Plugin\_tableWithSearch method*), [129](#)  
[IsConvex\(\)](#) (*in module optimeed.visualize.openGL.triangulate\_polygon*), [121](#)  
[isInitialized\(\)](#) (*Plugin\_listWithSearch method*), [127](#)  
[isInitialized\(\)](#) (*Plugin\_tableWithSearch method*), [129](#)  
[isNonePrintMessage\(\)](#) (*in module optimeed.core*), [50](#)  
[isNonePrintMessage\(\)](#) (*in module optimeed.core.tools*), [48](#)  
[isOnWindows](#) (*in module optimeed.visualize.graphs.pyqtgraphRedefine*), [92](#)  
[isScattered\(\)](#) (*Data method*), [38](#), [57](#), [143](#)
- ## J
- [json\\_to\\_obj\(\)](#) (*in module optimeed.core*), [52](#), [65](#)  
[json\\_to\\_obj\(\)](#) (*in module optimeed.core.myjson*), [43](#)  
[json\\_to\\_obj\\_safe\(\)](#) (*in module optimeed.core*), [52](#), [65](#)  
[json\\_to\\_obj\\_safe\(\)](#) (*in module optimeed.core.myjson*), [44](#)
- ## K
- [keyboard\\_push\\_action\(\)](#) (*DeviceDrawerInterface method*), [119](#), [121](#), [154](#)  
[keyboardPushAction\(\)](#) (*ContextHandler method*), [118](#)  
[keyboardReleaseAction\(\)](#) (*ContextHandler method*), [118](#)  
[keyPressEvent\(\)](#) (*MainWindow method*), [136](#), [141](#)  
[keyPressEvent\(\)](#) (*Widget\_graphsVisualLite method*), [98](#), [100](#), [147](#)  
[keyPressEvent\(\)](#) (*Widget\_openGL method*), [128](#), [131](#), [157](#)
- [KWARGS\\_ALGO](#) (*MultiObjective\_GA attribute*), [79](#), [80](#), [86](#)
- ## L
- [last\\_step\(\)](#) (*EvolutionaryConvergence method*), [71](#), [73](#)  
[launch\\_embarrassingly\\_parallel\\_sensitivity\(\)](#) (*in module optimeed consolidate*), [27](#)  
[launch\\_embarrassingly\\_parallel\\_sensitivity\(\)](#) (*in module optimeed consolidate.sensitivity\_analysis*), [22](#)  
[launch\\_optimization\(\)](#) (*OptimizationDisplayer method*), [133](#), [140](#)  
[leastSquare\(\)](#) (*in module optimeed consolidate*), [25](#)  
[leastSquare\(\)](#) (*in module optimeed consolidate.fit*), [19](#)  
[level\(\)](#) (*in module optimeed.core.ansi2html.style*), [29](#)  
[link\\_axes\(\)](#) (*Widget\_graphsVisualLite method*), [98](#), [100](#), [147](#)  
[LinkDataGraph](#) (*class in optimeed.core*), [60](#)  
[LinkDataGraph](#) (*class in optimeed.core.linkDataGraph*), [41](#)  
[linkify\(\)](#) (*in module optimeed.core.ansi2html.converter*), [28](#)  
[linkXToGraph\(\)](#) (*GraphVisual method*), [92](#)  
[linspace\(\)](#) (*in module optimeed.core*), [63](#)  
[linspace\(\)](#) (*in module optimeed.core.additional\_tools*), [31](#)  
[list\\_of\\_optimization\\_variables](#) (*SensitivityParameters attribute*), [21](#), [26](#)  
[ListDataStruct](#) (*class in optimeed consolidate*), [23](#)  
[ListDataStruct](#) (*class in optimeed.core*), [53](#)  
[ListDataStruct](#) (*class in optimeed.core.collection*), [33](#)  
[ListDataStruct\\_Interface](#) (*class in optimeed.core*), [53](#)  
[ListDataStruct\\_Interface](#) (*class in optimeed.core.collection*), [32](#)  
[log\\_after\\_evaluation\(\)](#) (*OptiHistoric method*), [82](#), [89](#)  
[log\\_mode\(\)](#) (*GraphVisual method*), [92](#)
- ## M
- [main\(\)](#) (*in module optimeed.core.ansi2html.converter*), [29](#)  
[MainWindow](#) (*class in optimeed.visualize*), [141](#)  
[MainWindow](#) (*class in optimeed.visualize.mainWindow*), [136](#)  
[map\\_index\(\)](#) (*\_AnimationTrace method*), [101](#)  
[map\\_vt100\\_box\\_code\(\)](#) (*in module optimeed.core.ansi2html.converter*), [28](#)  
[MaterialRenderingProperties](#) (*class in optimeed.visualize*), [154](#)



MaterialRenderingProperties (class in *optimeed.visualize.openGL*), 121  
 MaterialRenderingProperties (class in *optimeed.visualize.openGL.materials*), 119  
 MathsToPhysics (class in *optimeed.optimize*), 85  
 MathsToPhysics (class in *optimeed.optimize.mathsToPhysics*), 69  
 MathsToPhysics (class in *optimeed.optimize.mathsToPhysics.mathsToPhysics*), 68  
 matplotlib\_colormap\_to\_pg\_colormap() (in module *optimeed.visualize.graphs.colormap\_pyqtgraph*), 90  
 MaxTimeTerminationCondition (class in *optimeed.optimize.optiAlgorithms.multiObjective\_GA*), 77  
 merge() (*Graphs* method), 39, 59, 144  
 merge() (*ListDataStruct* method), 24, 33, 54  
 merge\_two\_dicts() (in module *optimeed.core*), 51  
 merge\_two\_dicts() (in module *optimeed.core.tools*), 49  
 MeshPlot (class in *optimeed.core*), 60  
 MeshPlot (class in *optimeed.core.graphs3*), 40  
 MeshPlot (class in *optimeed.visualize*), 146  
 meshPolygon() (in module *optimeed.visualize.openGL.triangulate\_polygon*), 121  
 minimumSizeHint() (*Widget\_openGL* method), 128, 131, 157  
 MODE\_LIGHT (in module *optimeed.visualize.openGL.contextHandler*), 118  
 MODE\_ROTATION (in module *optimeed.visualize.openGL.contextHandler*), 118  
 MODE\_ZOOM (in module *optimeed.visualize.openGL.contextHandler*), 118  
 modify\_paintElems() (*TraceVisual.\_ModifiedPaintElem* method), 95  
 MODULE\_TAG (in module *optimeed.core*), 64  
 MODULE\_TAG (in module *optimeed.core.myjson*), 43  
 Monobjective\_PSO (class in *optimeed.optimize*), 87  
 Monobjective\_PSO (class in *optimeed.optimize.optiAlgorithms*), 80  
 Monobjective\_PSO (class in *optimeed.optimize.optiAlgorithms.monobjective\_PSO*), 76  
 mouseClicAction() (*ContextHandler* method), 118  
 mouseMotionAction() (*ContextHandler* method), 118  
 mouseMoveEvent() (*Widget\_openGL* method), 128, 131, 157  
 mousePressEvent() (*Widget\_openGL* method), 128, 131, 157  
 mouseReleaseEvent() (*Widget\_menuButton* method), 128, 131, 157  
 mouseWheelAction() (*ContextHandler* method), 118  
 MultiList (class in *optimeed.optimize.optiAlgorithms.convergence.hypervolume*), 72  
 MultiList.Node (class in *optimeed.optimize.optiAlgorithms.convergence.hypervolume*), 72  
 MultiObjective\_GA (class in *optimeed.optimize*), 86  
 MultiObjective\_GA (class in *optimeed.optimize.optiAlgorithms*), 80  
 MultiObjective\_GA (class in *optimeed.optimize.optiAlgorithms.multiObjective\_GA*), 79  
 my\_callback() (*MyMultiprocessEvaluator* method), 79  
 my\_fft() (in module *optimeed.core*), 63  
 my\_fft() (in module *optimeed.core.additional\_tools*), 31  
 my\_fourier() (in module *optimeed.core*), 64  
 my\_fourier() (in module *optimeed.core.additional\_tools*), 31  
 myAxis (class in *optimeed.visualize.graphs.pyqtgraphRedefine*), 94  
 MyGenerator (class in *optimeed.optimize.optiAlgorithms.multiObjective\_GA*), 77  
 myGraphicsLayout (class in *optimeed.visualize.graphs.pyqtgraphRedefine*), 93  
 myGraphicsLayoutWidget (class in *optimeed.visualize.graphs.pyqtgraphRedefine*), 93  
 myItemSample (class in *optimeed.visualize.graphs.pyqtgraphRedefine*), 93  
 myLabelItem (class in *optimeed.visualize.graphs.pyqtgraphRedefine*), 94  
 myLegend (class in *optimeed.visualize.graphs.pyqtgraphRedefine*), 94  
 MyMapEvaluator (class in *optimeed.optimize.optiAlgorithms.multiObjective\_GA*), 78  
 MyMapEvaluator (class in *optimeed.optimize.optiAlgorithms.pyswarm*), 74  
 MyMapEvaluator (class in *optimeed.optimize.optiAlgorithms.pyswarm*), 74

- meed.optimize.optiAlgorithms.pyswarm.pso*), 73
- MyMultiprocessEvaluator* (class in *optimeed.optimize.optiAlgorithms.multiObjective\_GA*), 78
- MyMultiprocessEvaluator* (class in *optimeed.optimize.optiAlgorithms.pyswarm*), 74
- MyMultiprocessEvaluator* (class in *optimeed.optimize.optiAlgorithms.pyswarm.pso*), 73
- MyProblem* (class in *optimeed.optimize.optiAlgorithms.multiObjective\_GA*), 77
- MyText* (class in *optimeed.visualize.openGL.contextHandler*), 118
- myWindows* (in module *optimeed.visualize*), 145
- myWindows* (in module *optimeed.visualize.fastPlot*), 135
- ## N
- name* (*Option\_bool* attribute), 45, 62
- name* (*Option\_dict* attribute), 45, 63
- name* (*Option\_float* attribute), 45, 63
- name* (*Option\_int* attribute), 23, 45, 62
- name* (*Option\_str* attribute), 45, 62
- name()* (*Plugin\_listWithSearch* method), 127
- name()* (*Plugin\_tableWithSearch* method), 129
- new\_figure()* (*WindowHolders* method), 135, 145
- new\_plot()* (*\_PlotHolders* method), 135, 145
- new\_plot()* (*WindowHolders* method), 135, 145
- next\_frame()* (*AnimationGUI* method), 101
- NLOpt\_Algorithm* (class in *optimeed.optimize.optiAlgorithms.NLOpt\_Algorithm*), 75
- normalize()* (in module *optimeed.visualize.openGL.quaternions*), 121
- NUMBER\_OF\_CORES* (*Monobjective\_PSO* attribute), 76, 80, 87
- NUMBER\_OF\_CORES* (*MultiObjective\_GA* attribute), 79, 80, 86
- NUMBER\_OF\_CORES* (*Parametric\_analysis* attribute), 21, 25
- NUMBER\_OF\_MODES* (in module *optimeed.visualize.openGL.contextHandler*), 118
- ## O
- obj\_to\_json()* (in module *optimeed.core*), 52, 65
- obj\_to\_json()* (in module *optimeed.core.myjson*), 44
- objectives* (*OptiHistoric.\_pointData* attribute), 81, 89
- objectives\_per\_step* (*EvolutionaryConvergence* attribute), 71, 73
- on\_click()* (*Widget\_graphsVisualLite* method), 98, 99, 147
- on\_update\_signal()* (*Widget\_lineDrawer* method), 126, 130, 157
- Onclick\_animate* (class in *optimeed.visualize*), 148
- Onclick\_animate* (class in *optimeed.visualize.onclick*), 111
- Onclick\_animate* (class in *optimeed.visualize.onclick.onclick\_animate*), 104
- Onclick\_changeSymbol* (class in *optimeed.visualize*), 148
- Onclick\_changeSymbol* (class in *optimeed.visualize.onclick*), 111
- Onclick\_changeSymbol* (class in *optimeed.visualize.onclick.onclick\_changeSymbol*), 105
- Onclick\_copySomething* (class in *optimeed.visualize*), 149
- Onclick\_copySomething* (class in *optimeed.visualize.onclick*), 112
- Onclick\_copySomething* (class in *optimeed.visualize.onclick.onclick\_copySomething*), 105
- Onclick\_delete* (class in *optimeed.visualize*), 149
- Onclick\_delete* (class in *optimeed.visualize.onclick*), 112
- Onclick\_delete* (class in *optimeed.visualize.onclick.onclick\_delete*), 106
- Onclick\_exportCollection* (class in *optimeed.visualize*), 149
- Onclick\_exportCollection* (class in *optimeed.visualize.onclick*), 112
- Onclick\_exportCollection* (class in *optimeed.visualize.onclick.onclick\_exportCollection*), 106
- Onclick\_exportToTxt* (class in *optimeed.visualize*), 150
- Onclick\_exportToTxt* (class in *optimeed.visualize.onclick*), 113
- Onclick\_exportToTxt* (class in *optimeed.visualize.onclick.onclick\_exportToTxt*), 107
- Onclick\_exportTrace* (class in *optimeed.visualize*), 150
- Onclick\_exportTrace* (class in *optimeed.visualize.onclick*), 113
- Onclick\_exportTrace* (class in *optimeed.visualize.onclick.onclick\_exportTrace*), 107
- Onclick\_extractPareto* (class in *optimeed.visualize*), 150
- Onclick\_extractPareto* (class in *optimeed.visualize.onclick*), 113

Onclick_extractPareto (class in optimeed.visualize.onclick.onclick_extractPareto), 108	Onselect_newTrace (class in optimeed.visualize.selector.onselect_newTrace), 123
Onclick_measure (class in optimeed.visualize), 145, 151	Onselect_splitTrace (class in optimeed.visualize), 156
Onclick_measure (class in optimeed.visualize.onclick), 114	Onselect_splitTrace (class in optimeed.visualize.selector), 125
Onclick_measure (class in optimeed.visualize.onclick.onclick_measure), 108	Onselect_splitTrace (class in optimeed.visualize.selector.onselect_splitTrace), 124
Onclick_removeTrace (class in optimeed.visualize), 151	OnselectInterface (class in optimeed.visualize), 155
Onclick_removeTrace (class in optimeed.visualize.onclick), 114	OnselectInterface (class in optimeed.visualize.selector), 124
Onclick_removeTrace (class in optimeed.visualize.onclick.onclick_removeTrace), 109	OnselectInterface (class in optimeed.visualize.selector.onselectInterface), 122
Onclick_representDevice (class in optimeed.visualize), 148	OPTI_ALGORITHM (MultiObjective_GA attribute), 79, 80, 86
Onclick_representDevice (class in optimeed.visualize.onclick), 114	OptiHistoric (class in optimeed.optimize), 89
Onclick_representDevice (class in optimeed.visualize.onclick.onclick_representDevice), 109	OptiHistoric (class in optimeed.optimize.optiHistoric), 81
Onclick_representDevice.DataInformationVoptiHistoric (class in optimeed.visualize), 148	OptiHistoric._LogParams (class in optimeed.optimize.optiHistoric), 89
Onclick_representDevice.DataInformationVoptiHistoric._pointData (class in optimeed.visualize.onclick), 115	OptiHistoric._pointData (class in optimeed.optimize.optiHistoric), 81
Onclick_representDevice.DataInformationVoptiHistoric._pointData (class in optimeed.visualize.onclick.onclick_representDevice), 109	optimeed (module), 19
Onclick_tojson (class in optimeed.visualize), 151	optimeed consolidate (module), 19
Onclick_tojson (class in optimeed.visualize.onclick), 115	optimeed consolidate.fit (module), 19
Onclick_tojson (class in optimeed.visualize.onclick.onclick_tojson), 110	optimeed consolidate.parametric_analysis (module), 20
OnclickInterface (class in optimeed.visualize), 152	optimeed consolidate.sensitivity_analysis (module), 21
OnclickInterface (class in optimeed.visualize.onclick), 115	optimeed consolidate.sensitivity_analysis_evaluation (module), 23
OnclickInterface (class in optimeed.visualize.onclick.onclickInterface), 104	optimeed.core (module), 27
Onselect_highlight (class in optimeed.visualize), 155	optimeed.core.additional_tools (module), 30
Onselect_highlight (class in optimeed.visualize.selector), 124	optimeed.core.ansi2html (module), 27
Onselect_highlight (class in optimeed.visualize.selector.onselect_highlight), 123	optimeed.core.ansi2html.converter (module), 27
Onselect_newTrace (class in optimeed.visualize), 155	optimeed.core.ansi2html.style (module), 29
Onselect_newTrace (class in optimeed.visualize.selector), 125	optimeed.core.ansi2html.util (module), 30
	optimeed.core.collection (module), 32
	optimeed.core.color_palette (module), 35
	optimeed.core.commonImport (module), 35
	optimeed.core.graphs (module), 36
	optimeed.core.graphs3 (module), 40
	optimeed.core.inkscape_manager (module), 41

optimeed.core.linkDataGraph (module), 41  
 optimeed.core.myjson (module), 43  
 optimeed.core.options (module), 44  
 optimeed.core.tikzTranslator (module), 46  
 optimeed.core.tools (module), 47  
 optimeed.optimize (module), 66  
 optimeed.optimize.characterization (module), 67  
 optimeed.optimize.characterization.characterization (module), 67  
 optimeed.optimize.characterization.interfaceCharacterization (module), 67  
 optimeed.optimize.mathsToPhysics (module), 68  
 optimeed.optimize.mathsToPhysics.interfaceMathsToPhysics (module), 68  
 optimeed.optimize.mathsToPhysics.mathsToPhysics (module), 68  
 optimeed.optimize.objAndCons (module), 69  
 optimeed.optimize.objAndCons.fastObjCons (module), 69  
 optimeed.optimize.objAndCons.interfaceObjCons (module), 70  
 optimeed.optimize.optiAlgorithms (module), 70  
 optimeed.optimize.optiAlgorithms.algorithmInterface (module), 76  
 optimeed.optimize.optiAlgorithms.convergence (module), 70  
 optimeed.optimize.optiAlgorithms.convergenceArrayConvergence (module), 70  
 optimeed.optimize.optiAlgorithms.convergenceHyperVolume (module), 71  
 optimeed.optimize.optiAlgorithms.convergenceInterfaceConvergence (module), 72  
 optimeed.optimize.optiAlgorithms.monobjectivePSO (module), 76  
 optimeed.optimize.optiAlgorithms.multiObjectivePSO (module), 77  
 optimeed.optimize.optiAlgorithms.NLOpt\_AdaptivePSO (module), 75  
 optimeed.optimize.optiAlgorithms.pyswarm (module), 73  
 optimeed.optimize.optiAlgorithms.pyswarmPSO (module), 73  
 optimeed.optimize.optiHistoric (module), 81  
 optimeed.optimize.optimizer (module), 84  
 optimeed.optimize.optiVariable (module), 82  
 optimeed.visualize (module), 90  
 optimeed.visualize.displayCollections (module), 132  
 optimeed.visualize.displayOptimization (module), 133  
 optimeed.visualize.displaySensitivity (module), 134  
 optimeed.visualize.fastPlot (module), 135  
 optimeed.visualize.fastPlot3 (module), 136  
 optimeed.visualize.graphs (module), 90  
 optimeed.visualize.graphs.colormap\_pyqtgraph (module), 90  
 optimeed.visualize.graphs.graphVisual (module), 90  
 optimeed.visualize.graphs.pyqtgraphRedefine (module), 92  
 optimeed.visualize.graphs.traceVisual (module), 95  
 optimeed.visualize.graphs.widget\_graphsVisual (module), 97  
 optimeed.visualize.mainWindow (module), 136  
 optimeed.visualize.onclick (module), 100  
 optimeed.visualize.onclick.animation\_examples (module), 102  
 optimeed.visualize.onclick.animationGUI (module), 100  
 optimeed.visualize.onclick.collectionExporterGUI (module), 103  
 optimeed.visualize.onclick.onclick\_animate (module), 104  
 optimeed.visualize.onclick.onclick\_changeSymbol (module), 105  
 optimeed.visualize.onclick.onclick\_copySomething (module), 105  
 optimeed.visualize.onclick.onclick\_delete (module), 106  
 optimeed.visualize.onclick.onclick\_exportCollection (module), 106  
 optimeed.visualize.onclick.onclick\_exportToTxt (module), 107  
 optimeed.visualize.onclick.onclick\_exportTrace (module), 107  
 optimeed.visualize.onclick.onclick\_extractPareto (module), 108  
 optimeed.visualize.onclick.onclick\_measure (module), 108  
 optimeed.visualize.onclick.onclick\_removeTrace (module), 109  
 optimeed.visualize.onclick.onclick\_representDevice (module), 109  
 optimeed.visualize.onclick.onclick\_tojson (module), 110  
 optimeed.visualize.onclick.onclickInterface (module), 104  
 optimeed.visualize.onclick.representDevice\_examples (module), 110  
 optimeed.visualize.openGL (module), 118



optimeed.visualize.openGL.contextHandler (module), 118  
 optimeed.visualize.openGL.deviceDrawerInterface (module), 119  
 optimeed.visualize.openGL.materials (module), 119  
 optimeed.visualize.openGL.openGL\_library (module), 120  
 optimeed.visualize.openGL.quaternions (module), 121  
 optimeed.visualize.openGL.triangulate\_polygon (module), 121  
 optimeed.visualize.process\_mainloop (module), 136  
 optimeed.visualize.selector (module), 122  
 optimeed.visualize.selector.onselect\_highlights (module), 122  
 optimeed.visualize.selector.onselect\_newPatterns (module), 123  
 optimeed.visualize.selector.onselect\_splitPlane (module), 124  
 optimeed.visualize.selector.onselectInterference (module), 122  
 optimeed.visualize.viewOptimizationResults (module), 137  
 optimeed.visualize.widgets (module), 125  
 optimeed.visualize.widgets.widget\_doubleSlider (module), 125  
 optimeed.visualize.widgets.widget\_image (module), 126  
 optimeed.visualize.widgets.widget\_lineDrawer (module), 126  
 optimeed.visualize.widgets.widget\_listWithSearch (module), 127  
 optimeed.visualize.widgets.widget\_listWithSearchPlugin (module), 127  
 optimeed.visualize.widgets.widget\_menuButton (module), 128  
 optimeed.visualize.widgets.widget\_openGL (module), 128  
 optimeed.visualize.widgets.widget\_tableWithSearch (module), 128  
 optimeed.visualize.widgets.widget\_tableWithSearchPlugin (module), 129  
 optimeed.visualize.widgets.widget\_text (module), 130  
 OptimizationDisplayer (class in optimeed.visualize), 139  
 OptimizationDisplayer (class in optimeed.visualize.displayOptimization), 133  
 OptimizationVariable (class in optimeed.optimize.optiVariable), 82  
 OptimizerSettings (class in optimeed.optimize), 89  
 OptimizerSettings (class in optimeed.optimize.optimizer), 84  
 Option\_bool (class in optimeed.core), 62  
 Option\_bool (class in optimeed.core.options), 44  
 Option\_class (class in optimeed consolidate), 23  
 Option\_class (class in optimeed.core), 63  
 Option\_class (class in optimeed.core.options), 45  
 Option\_dict (class in optimeed.core), 63  
 Option\_dict (class in optimeed.core.options), 45  
 Option\_float (class in optimeed.core), 62  
 Option\_float (class in optimeed.core.options), 45  
 Option\_int (class in optimeed consolidate), 23  
 Option\_int (class in optimeed.core), 62  
 Option\_int (class in optimeed.core.options), 45  
 Option\_str (class in optimeed.core), 62  
 Option\_str (class in optimeed.core.options), 45  
 options\_bool (Option\_class attribute), 23, 45, 63  
 options\_dict (Option\_class attribute), 23, 45, 63  
 options\_float (Option\_class attribute), 23, 45, 63  
 options\_int (Option\_class attribute), 23, 45, 63  
 options\_str (Option\_class attribute), 23, 45, 63  
 order\_lists() (in module optimeed.core), 51  
 order\_lists() (in module optimeed.core.tools), 49  
 P  
 paint() (\_LineItem method), 108  
 paint() (myItemSample method), 94  
 painter() (myLegend method), 94  
 paintEvent() (Widget\_lineDrawer method), 127, 130, 157  
 paintGL() (Widget\_openGL method), 128, 131, 157  
 param\_values (SensitivityParameters attribute), 21, 26  
 Parametric\_analysis (class in optimeed consolidate), 25  
 Parametric\_analysis (class in optimeed consolidate.parametric\_analysis), 21  
 Parametric\_Collection (class in optimeed consolidate), 25  
 Parametric\_Collection (class in optimeed consolidate.parametric\_analysis), 20  
 Parametric\_minmax (class in optimeed consolidate), 25  
 Parametric\_minmax (class in optimeed consolidate.parametric\_analysis), 20  
 Parametric\_parameter (class in optimeed consolidate), 25  
 Parametric\_parameter (class in optimeed consolidate.parametric\_analysis), 20  
 paramsToEvaluate (SensitivityResults attribute), 21

paretos\_per\_step (*EvolutionaryConvergence* attribute), 71, 73  
 partition() (in module *optimeed.core*), 64  
 partition() (in module *optimeed.core.additional\_tools*), 31  
 pause\_play() (*AnimationGUI* method), 101  
 Performance\_ListDataStruct (class in *optimeed.core*), 54  
 Performance\_ListDataStruct (class in *optimeed.core.collection*), 34  
 Pink\_material (in module *optimeed.visualize*), 155  
 Pink\_material (in module *optimeed.visualize.openGL*), 122  
 Pink\_material (in module *optimeed.visualize.openGL.materials*), 120  
 plot() (in module *optimeed.visualize*), 145  
 plot() (in module *optimeed.visualize.fastPlot*), 135  
 Plot3D\_Generic (class in *optimeed.core*), 59  
 Plot3D\_Generic (class in *optimeed.core.graphs3*), 40  
 Plugin\_listWithSearch (class in *optimeed.visualize.widgets.widget\_listWithSearchplugin*), 127  
 Plugin\_tableWithSearch (class in *optimeed.visualize.widgets.widget\_tableWithSearchplugin*), 129  
 pol2cart() (in module *optimeed.core*), 63  
 pol2cart() (in module *optimeed.core.additional\_tools*), 31  
 POPULATION\_SIZE (*NLOpt\_Algorithm* attribute), 75  
 prepare() (*Ansi2HTMLConverter* method), 29, 30  
 prepare\_embarrassingly\_parallel\_sensitivity() (in module *optimeed consolidate*), 27  
 prepare\_embarrassingly\_parallel\_sensitivity() (in module *optimeed consolidate.sensitivity\_analysis*), 22  
 preProcess() (*HyperVolume* method), 72  
 printIfShown() (in module *optimeed.core*), 50, 52, 55, 64, 66  
 printIfShown() (in module *optimeed.core.tools*), 48  
 printIfShown() (in module *optimeed.visualize*), 146  
 process\_qt\_events() (in module *optimeed.visualize.process\_mainloop*), 137  
 produce\_headers() (*Ansi2HTMLConverter* method), 29, 30  
 pso() (in module *optimeed.optimize.optiAlgorithms.pyswarm*), 74  
 pso() (in module *optimeed.optimize.optiAlgorithms.pyswarm.pso*), 74  
 PURPLE (*text\_format* attribute), 47, 49

## Q

q\_conjugate() (in module *optimeed.visualize.openGL.quaternions*), 121  
 q\_mult() (in module *optimeed.visualize.openGL.quaternions*), 121  
 q\_to\_axisangle() (in module *optimeed.visualize.openGL.quaternions*), 121  
 q\_to\_mat4() (in module *optimeed.visualize.openGL.quaternions*), 121  
 quicksort() (in module *optimeed.core*), 64  
 quicksort() (in module *optimeed.core.additional\_tools*), 31  
 qv\_mult() (in module *optimeed.visualize.openGL.quaternions*), 121

## R

r\_squared() (in module *optimeed consolidate.fit*), 20  
 read\_to\_unicode() (in module *optimeed.core.ansi2html.util*), 30  
 Real\_OptimizationVariable (class in *optimeed.optimize*), 88  
 Real\_OptimizationVariable (class in *optimeed.optimize.optiVariable*), 82  
 reconstitute\_signal() (in module *optimeed.core*), 63  
 reconstitute\_signal() (in module *optimeed.core.additional\_tools*), 31  
 RED (*text\_format* attribute), 47, 49  
 Red\_material (in module *optimeed.visualize*), 155  
 Red\_material (in module *optimeed.visualize.openGL*), 122  
 Red\_material (in module *optimeed.visualize.openGL.materials*), 120  
 redraw() (*ContextHandler* method), 118  
 reevaluate\_solutions() (*\_Evaluator* method), 85  
 reformatXYtoList() (in module *optimeed.visualize.openGL.triangulate\_polygon*), 121  
 refreshTraceList() (*Widget\_graphsVisual* method), 99, 100, 141, 148  
 reinsert() (*MultiList* method), 72  
 remove() (*MultiList* method), 72  
 remove\_collection() (*CollectionDisplay* method), 132, 138  
 remove\_collection() (*LinkDataGraph* method), 41, 61  
 remove\_feature() (*GraphVisual* method), 91  
 remove\_forced\_hide\_row() (*Widget\_tableWithSearch* method), 129, 131, 158  
 remove\_graph() (*Graphs* method), 39, 59, 144  
 remove\_module\_tree\_from\_string() (in module *optimeed.core*), 52, 66

remove\_module\_tree\_from\_string() (in module *optimeed.core.myjson*), 44  
 remove\_trace() (Graph method), 38, 58  
 remove\_trace() (Graphs method), 39, 59, 144  
 reorder() (Performance\_ListDataStruct method), 34, 55  
 Represent\_brut\_attributes (class in *optimeed.visualize*), 152  
 Represent\_brut\_attributes (class in *optimeed.visualize.onclick*), 116  
 Represent\_brut\_attributes (class in *optimeed.visualize.onclick.representDevice\_examples*), 111  
 Represent\_image (class in *optimeed.visualize*), 152  
 Represent\_image (class in *optimeed.visualize.onclick*), 115  
 Represent\_image (class in *optimeed.visualize.onclick.representDevice\_examples*), 111  
 Represent\_lines (class in *optimeed.visualize*), 152  
 Represent\_lines (class in *optimeed.visualize.onclick*), 116  
 Represent\_lines (class in *optimeed.visualize.onclick.representDevice\_examples*), 110  
 Represent\_opengl (class in *optimeed.visualize*), 152  
 Represent\_opengl (class in *optimeed.visualize.onclick*), 115  
 Represent\_opengl (class in *optimeed.visualize.onclick.representDevice\_examples*), 111  
 Represent\_txt\_function (class in *optimeed.visualize*), 152  
 Represent\_txt\_function (class in *optimeed.visualize.onclick*), 116  
 Represent\_txt\_function (class in *optimeed.visualize.onclick.representDevice\_examples*), 110  
 RepresentDeviceInterface (class in *optimeed.visualize*), 148  
 RepresentDeviceInterface (class in *optimeed.visualize.onclick*), 111  
 RepresentDeviceInterface (class in *optimeed.visualize.onclick.onclick\_representDevice*), 109  
 reset() (PlotHolders method), 135, 145  
 reset() (State method), 28  
 reset() (AlgorithmInterface method), 76  
 reset() (CollectionExporterGUI method), 104  
 reset() (Graphs method), 40, 59, 144  
 reset() (MultiObjective\_GA method), 79, 80, 87  
 reset() (TraceVisual.\_ModifiedPaintElem method), 95  
 reset\_all() (AnimationGUI method), 101  
 reset\_all\_brushes() (TraceVisual method), 96  
 reset\_all\_symbolPens() (TraceVisual method), 97  
 reset\_brush() (TraceVisual method), 96  
 reset\_brushes() (TraceVisual method), 96  
 reset\_data() (ListDataStruct method), 24, 33, 54  
 reset\_distance() (OnClick\_measure method), 109, 114, 145, 151  
 reset\_graph() (OnClick\_exportCollection method), 106, 113, 150  
 reset\_paintElem() (TraceVisual.\_ModifiedPaintElem method), 95  
 reset\_symbol() (TraceVisual method), 96  
 reset\_symbolPen() (TraceVisual method), 97  
 reset\_symbolPens() (TraceVisual method), 97  
 resizeEvent() (myAxis method), 95  
 resizeGL() (Widget\_openGL method), 128, 131, 157  
 resizeWindowAction() (ContextHandler method), 118  
 rgetattr() (in module *optimeed consolidate*), 25  
 rgetattr() (in module *optimeed.core*), 50, 52, 64  
 rgetattr() (in module *optimeed.core.tools*), 48  
 rsetattr() (in module *optimeed consolidate*), 25  
 rsetattr() (in module *optimeed.core*), 50, 64  
 rsetattr() (in module *optimeed.core.tools*), 48  
 Rule (class in *optimeed.core.ansi2html.style*), 29  
 run() (AnimationGUI method), 102  
 run() (MainWindow method), 136, 141  
 run() (Parametric\_analysis method), 21, 25  
 run\_optimization() (in module *optimeed.optimize*), 89  
 run\_optimization() (in module *optimeed.optimize.optimizer*), 85  
 run\_optimization\_displayer() (in module *optimeed.visualize.displayOptimization*), 133

## S

save() (AutosaveStruct method), 25, 32, 53  
 save() (ListDataStruct method), 24, 33, 53  
 save() (OptiHistoric method), 82, 89  
 save() (Performance\_ListDataStruct method), 34, 55  
 SaveableObject (class in *optimeed.core*), 65  
 SaveableObject (class in *optimeed.core.myjson*), 43  
 ScatterPlot3 (class in *optimeed.core*), 60  
 ScatterPlot3 (class in *optimeed.core.graphs3*), 40  
 ScatterPlot3 (class in *optimeed.visualize*), 146  
 SCHEME (in module *optimeed.core.ansi2html.style*), 29  
 select\_folder\_and\_export() (Widget\_graphsVisualLite method), 98, 99, 147  
 selector\_updated() (Onselect\_highlight method), 123, 124, 155  
 selector\_updated() (Onselect\_newTrace method), 123, 125, 155  
 selector\_updated() (Onselect\_splitTrace method), 124, 125, 156

SensitivityDisplayer (class in *optimeed.visualize*), 138  
 SensitivityDisplayer (class in *optimeed.visualize.displaySensitivity*), 134  
 SensitivityParameters (class in *optimeed consolidate*), 26  
 SensitivityParameters (class in *optimeed consolidate.sensitivity\_analysis*), 21  
 SensitivityResults (class in *optimeed consolidate.sensitivity\_analysis*), 21  
 sequence (in module *optimeed.visualize.graphs.colormap\_pyqtgraph*), 90  
 set\_action\_selector() (*CollectionDisplayer method*), 132, 138  
 set\_actionOnClick() (*Widget\_graphsVisualLite method*), 98, 100, 147  
 set\_actionOnClose() (*MainWindow method*), 136, 141  
 set\_actions\_on\_click() (*CollectionDisplayer method*), 132, 138  
 set\_actions\_on\_click() (*Widget\_graphsVisual method*), 99, 100, 141, 148  
 set\_actionsOnClick() (*OptimizationDisplayer method*), 133, 139  
 set\_brush() (*TraceVisual method*), 96  
 set\_brushes() (*TraceVisual method*), 96  
 set\_collection() (*CollectionExporterGUI method*), 104  
 set\_color() (*Data method*), 38, 58, 143  
 set\_color() (*TraceVisual method*), 95  
 set\_color\_palette() (*GraphVisual method*), 91  
 set\_convergence() (*OptiHistoric method*), 82, 89  
 set\_curr\_brush() (*\_AnimationTrace method*), 101  
 set\_curr\_step() (*EvolutionaryConvergence method*), 71, 73  
 set\_currFigure() (*WindowHolders method*), 135, 145  
 set\_data() (*Data method*), 36, 56, 141  
 set\_data() (*ListDataStruct method*), 24, 33, 54  
 set\_data\_at\_index() (*ListDataStruct method*), 24, 33, 54  
 set\_data\_at\_index() (*Performance\_ListDataStruct method*), 34, 55  
 set\_data\_at\_indices() (*Performance\_ListDataStruct method*), 34, 55  
 set\_deviceDrawer() (*ContextHandler method*), 118  
 set\_deviceDrawer() (*Widget\_openGL method*), 128, 131, 157  
 set\_deviceToDraw() (*ContextHandler method*), 118  
 set\_deviceToDraw() (*Widget\_openGL method*), 128, 131, 157  
 set\_entries() (*Widget\_tableWithSearch method*), 129, 131, 158  
 set\_evaluationFunction() (*Monobjective\_PSO method*), 77, 81, 87  
 set\_evaluationFunction() (*MultiObjective\_GA method*), 79, 80, 86  
 set\_evaluationFunction() (*NLOpt\_Algorithm method*), 75  
 set\_filename() (*AutosaveStruct method*), 24, 32, 53  
 set\_font() (*myLegend method*), 94  
 set\_fontLabel() (*GraphVisual method*), 91  
 set\_fontLegend() (*GraphVisual method*), 91  
 set\_fontTicks() (*GraphVisual method*), 90  
 set\_graph\_disposition() (*myGraphicsLayout method*), 93  
 set\_graph\_disposition() (*Widget\_graphsVisualLite method*), 97, 99, 146  
 set\_graph\_properties() (*GraphVisual method*), 91  
 set\_idle\_brush() (*\_AnimationTrace method*), 101  
 set\_image() (*Widget\_image method*), 126, 130, 156  
 set\_indices\_points\_to\_plot() (*Data method*), 38, 58, 143  
 set\_item() (*Widget\_tableWithSearch method*), 129, 131, 158  
 set\_kwargs() (*Data method*), 36, 56, 141  
 set\_label\_pos() (*GraphVisual method*), 91  
 set\_label\_pos() (*myAxis method*), 95  
 set\_legend() (*Data method*), 38, 58, 143  
 set\_legend() (*GraphVisual method*), 91  
 set\_lims() (*GraphVisual method*), 91  
 set\_lines() (*Widget\_lineDrawer method*), 126, 130, 157  
 set\_list() (*Widget\_listWithSearch method*), 127, 130, 156  
 set\_maxtime() (*Monobjective\_PSO method*), 77, 81, 87  
 set\_maxtime() (*MultiObjective\_GA method*), 79, 80, 87  
 set\_maxtime() (*NLOpt\_Algorithm method*), 76  
 set\_meta() (*Data method*), 36, 56, 141  
 set\_number\_ticks() (*myAxis method*), 95  
 set\_numberTicks() (*GraphVisual method*), 91  
 set\_offset() (*myItemSample method*), 94  
 set\_offset\_sample() (*myLegend method*), 94  
 set\_option() (*Option\_class method*), 23, 45, 63  
 set\_permutations() (*Data method*), 37, 57, 143  
 set\_pop\_size() (*ConvergenceManager method*), 75  
 set\_position() (*myLegend method*), 94  
 set\_refreshTime() (*AnimationGUI method*), 102  
 set\_results() (*OptiHistoric method*), 82, 89  
 set\_shadow() (*CollectionDisplayer method*), 132, 138



`set_shadow_collection()` (*LinkDataGraph method*), 42, 61  
`set_space_sample_label()` (*myLegend method*), 94  
`set_specialButtonsMapping()` (*ContextHandler method*), 118  
`set_symbol()` (*TraceVisual method*), 96  
`set_symbolPen()` (*TraceVisual method*), 97  
`set_symbolPens()` (*TraceVisual method*), 97  
`set_text()` (*Widget\_text method*), 130, 132, 158  
`set_text()` (*Widget\_text\_scrollable method*), 130, 132, 158  
`set_title()` (*\_PlotHolders method*), 135, 145  
`set_title()` (*GraphVisual method*), 91  
`set_title()` (*in module optimeed.visualize*), 146  
`set_title()` (*in module optimeed.visualize.fastPlot*), 136  
`set_title()` (*Widget\_graphsVisualLite method*), 98, 100, 147  
`set_title()` (*WindowHolders method*), 135, 145  
`set_value()` (*Base\_Option method*), 44, 62  
`set_value()` (*Option\_bool method*), 45, 62  
`set_value()` (*Option\_dict method*), 45, 63  
`set_value()` (*Option\_float method*), 45, 63  
`set_value()` (*Option\_int method*), 23, 45, 62  
`set_value()` (*Option\_str method*), 45, 62  
`set_width_cell()` (*myItemSample method*), 94  
`set_width_cell_sample()` (*myLegend method*), 94  
`setCurrentShow()` (*in module optimeed.core*), 51  
`setCurrentShow()` (*in module optimeed.core.commonImport*), 35  
`setMaximum()` (*widget\_doubleSlider method*), 126  
`setMinimum()` (*widget\_doubleSlider method*), 126  
`setPath_workspace()` (*in module optimeed.core*), 50  
`setPath_workspace()` (*in module optimeed.core.tools*), 48  
`setSingleStep()` (*widget\_doubleSlider method*), 126  
`setText()` (*myLabelItem method*), 94  
`setValue()` (*widget\_doubleSlider method*), 126  
`SeveralTerminationCondition` (*class in optimeed.optimize.optiAlgorithms.multiObjective\_GA*), 78  
`shouldTerminate()` (*ConvergenceTerminationCondition method*), 78  
`shouldTerminate()` (*MaxTimeTerminationCondition method*), 78  
`shouldTerminate()` (*SeveralTerminationCondition method*), 78  
`show()` (*in module optimeed.visualize*), 146  
`show()` (*in module optimeed.visualize.fastPlot*), 135  
`show()` (*TraceVisual method*), 96  
`show()` (*WindowHolders method*), 135, 145  
`show_all()` (*\_AnimationTrace method*), 101  
`show_all()` (*AnimationGUI method*), 101  
`SHOW_CONSTRAINTS` (*OptimizationDisplayer attribute*), 133, 139  
`SHOW_CURRENT` (*in module optimeed.core*), 51  
`SHOW_CURRENT` (*in module optimeed.core.commonImport*), 35  
`SHOW_DEBUG` (*in module optimeed.core*), 51, 52  
`SHOW_DEBUG` (*in module optimeed.core.commonImport*), 35  
`SHOW_ERROR` (*in module optimeed.core*), 51, 52, 64  
`SHOW_ERROR` (*in module optimeed.core.commonImport*), 35  
`SHOW_INFO` (*in module optimeed.core*), 51, 52  
`SHOW_INFO` (*in module optimeed.core.commonImport*), 35  
`SHOW_LOGS` (*in module optimeed.core*), 51  
`SHOW_LOGS` (*in module optimeed.core.commonImport*), 35  
`SHOW_WARNING` (*in module optimeed.core*), 51, 52, 55, 64, 66  
`SHOW_WARNING` (*in module optimeed.core.commonImport*), 35  
`SHOW_WARNING` (*in module optimeed.visualize*), 146  
`showEvent()` (*Widget\_menuButton method*), 128, 131, 157  
`showRow()` (*Widget\_tableWithSearch method*), 129, 131, 157  
`signal_graph_changed` (*Widget\_graphsVisualLite attribute*), 97, 99, 146  
`signal_has_exported` (*CollectionExporterGUI attribute*), 103  
`signal_has_reset` (*CollectionExporterGUI attribute*), 104  
`signal_must_update` (*TraceVisual attribute*), 95  
`signal_must_update` (*Widget\_graphsVisualLite attribute*), 97, 99, 146  
`signal_must_update` (*Widget\_lineDrawer attribute*), 126, 130, 157  
`signal_optimization_over` (*OptimizationDisplayer attribute*), 133, 139  
`Silver_material` (*in module optimeed.visualize*), 155  
`Silver_material` (*in module optimeed.visualize.openGL*), 122  
`Silver_material` (*in module optimeed.visualize.openGL.materials*), 119  
`SingleObjectSaveLoad` (*class in optimeed.core*), 52  
`SingleObjectSaveLoad` (*class in optimeed.core.collection*), 32  
`singleStep()` (*widget\_doubleSlider method*), 126  
`sizeHint()` (*Widget\_openGL method*), 128, 131, 157

[slider\\_handler\(\)](#) (*AnimationGUI method*), 101  
[SLIDER\\_MAXIMUM\\_VALUE](#) (*AnimationGUI attribute*), 101  
[SLIDER\\_MINIMUM\\_VALUE](#) (*AnimationGUI attribute*), 101  
[software\\_version\(\)](#) (*in module optimeed.core*), 49  
[software\\_version\(\)](#) (*in module optimeed.core.tools*), 47  
[sortByDimension\(\)](#) (*HyperVolume method*), 72  
[sparse\\_subset\(\)](#) (*in module optimeed.core*), 64  
[sparse\\_subset\(\)](#) (*in module optimeed.core.additional\_tools*), 31  
[SpecialButtonsMapping](#) (*class in optimeed.visualize.openGL.contextHandler*), 118  
[start\(\)](#) (*\_Evaluator method*), 84  
[start\(\)](#) (*OptiHistoric method*), 82, 90  
[start\\_autorefresh\(\)](#) (*OptimizationDisplayer method*), 133, 140  
[start\\_autosave\(\)](#) (*AutosaveStruct method*), 25, 32, 53  
[start\\_qt\\_mainloop\(\)](#) (*in module optimeed.visualize*), 141  
[start\\_qt\\_mainloop\(\)](#) (*in module optimeed.visualize.process\_mainloop*), 137  
[Steel\\_material](#) (*in module optimeed.visualize*), 155  
[Steel\\_material](#) (*in module optimeed.visualize.openGL*), 122  
[Steel\\_material](#) (*in module optimeed.visualize.openGL.materials*), 119  
[stop\\_autorefresh\(\)](#) (*OptimizationDisplayer method*), 133, 140  
[stop\\_autosave\(\)](#) (*AutosaveStruct method*), 24, 32, 53  
[stop\\_qt\\_mainloop\(\)](#) (*in module optimeed.visualize.process\_mainloop*), 137  
[str\\_all\\_attr\(\)](#) (*in module optimeed.core*), 50  
[str\\_all\\_attr\(\)](#) (*in module optimeed.core.tools*), 48  
[success](#) (*SensitivityResults attribute*), 21  
[SurfPlot](#) (*class in optimeed.core*), 60  
[SurfPlot](#) (*class in optimeed.core.graphs3*), 40  
[SurfPlot](#) (*class in optimeed.visualize*), 146  
[symbol\\_isfilled\(\)](#) (*Data method*), 36, 56, 141

## T

[templates\\_tikz](#) (*in module optimeed.core.tikzTranslator*), 46  
[text\\_format](#) (*class in optimeed.core*), 49  
[text\\_format](#) (*class in optimeed.core.tools*), 47  
[theLock](#) (*in module optimeed.core*), 54  
[theLock](#) (*in module optimeed.core.collection*), 34  
[time](#) (*OptiHistoric.\_pointData attribute*), 81, 89  
[to\\_css\\_classes\(\)](#) (*\_State method*), 28  
[toggle\(\)](#) (*TraceVisual method*), 96

[toolTip\(\)](#) (*Plugin\_listWithSearch method*), 127  
[toolTip\(\)](#) (*Plugin\_tableWithSearch method*), 129  
[TraceVisual](#) (*class in optimeed.visualize.graphs.traceVisual*), 95  
[TraceVisual.\\_ModifiedPaintElem](#) (*class in optimeed.visualize.graphs.traceVisual*), 95  
[truncate\(\)](#) (*in module optimeed.core*), 50  
[truncate\(\)](#) (*in module optimeed.core.tools*), 48

## U

[UNDERLINE](#) (*text\_format attribute*), 47, 49  
[universalPath\(\)](#) (*in module optimeed.core*), 50  
[universalPath\(\)](#) (*in module optimeed.core.tools*), 48  
[update\(\)](#) (*GraphVisual method*), 92  
[update\\_graphs\(\)](#) (*CollectionDisplayer method*), 132, 138  
[update\\_graphs\(\)](#) (*LinkDataGraph method*), 42, 61  
[update\\_graphs\(\)](#) (*Widget\_graphsVisualLite method*), 98, 99, 147  
[update\\_label](#) (*myAxis attribute*), 94  
[update\\_widget\\_w\\_animation\(\)](#) (*Animate\_lines method*), 103, 116, 153  
[update\\_widget\\_w\\_animation\(\)](#) (*Animate\_lines\_and\_text method*), 103, 117, 154  
[update\\_widget\\_w\\_animation\(\)](#) (*Animate\_openGL method*), 102, 117, 153  
[update\\_widget\\_w\\_animation\(\)](#) (*Animate\_openGL\_and\_text method*), 102, 117, 154  
[updateChildren\(\)](#) (*Graphs method*), 38, 58, 143  
[updateSize\(\)](#) (*myLegend method*), 94  
[updateTrace\(\)](#) (*TraceVisual method*), 96  
[useOpenGL\(\)](#) (*myGraphicsLayoutWidget method*), 93

## V

[val\\_max](#) (*Integer\_OptimizationVariable attribute*), 83, 88  
[val\\_max](#) (*Real\_OptimizationVariable attribute*), 83, 88  
[val\\_min](#) (*Integer\_OptimizationVariable attribute*), 83, 88  
[val\\_min](#) (*Real\_OptimizationVariable attribute*), 83, 88  
[value](#) (*Option\_bool attribute*), 45, 62  
[value](#) (*Option\_dict attribute*), 45, 63  
[value](#) (*Option\_float attribute*), 45, 63  
[value](#) (*Option\_int attribute*), 23, 45, 62  
[value](#) (*Option\_str attribute*), 45, 62  
[value\(\)](#) (*widget\_doubleSlider method*), 126  
[VERSION](#) (*in module optimeed*), 158  
[ViewOptimizationResults](#) (*class in optimeed.visualize*), 140  
[ViewOptimizationResults](#) (*class in optimeed.visualize.viewOptimizationResults*),

137  
 VT100\_BOX\_CODES (in module *optimeed.core.ansi2html.converter*), 28

## W

*whatsThis()* (*Plugin\_listWithSearch* method), 127  
*whatsThis()* (*Plugin\_tableWithSearch* method), 129  
*wheelEvent()* (*Widget\_openGL* method), 128, 131, 157  
 WHITE (*text\_format* attribute), 47, 49  
*widget\_doubleSlider* (class in *optimeed.visualize.widgets.widget\_doubleSlider*), 126  
*Widget\_graphsVisual* (class in *optimeed.visualize*), 141, 148  
*Widget\_graphsVisual* (class in *optimeed.visualize.graphs*), 100  
*Widget\_graphsVisual* (class in *optimeed.visualize.graphs.widget\_graphsVisual*), 98  
*Widget\_graphsVisualLite* (class in *optimeed.visualize*), 146  
*Widget\_graphsVisualLite* (class in *optimeed.visualize.graphs*), 99  
*Widget\_graphsVisualLite* (class in *optimeed.visualize.graphs.widget\_graphsVisual*), 97  
*Widget\_image* (class in *optimeed.visualize*), 156  
*Widget\_image* (class in *optimeed.visualize.widgets*), 130  
*Widget\_image* (class in *optimeed.visualize.widgets.widget\_image*), 126  
*Widget\_lineDrawer* (class in *optimeed.visualize*), 156  
*Widget\_lineDrawer* (class in *optimeed.visualize.widgets*), 130  
*Widget\_lineDrawer* (class in *optimeed.visualize.widgets.widget\_lineDrawer*), 126  
*Widget\_listWithSearch* (class in *optimeed.visualize*), 156  
*Widget\_listWithSearch* (class in *optimeed.visualize.widgets*), 130  
*Widget\_listWithSearch* (class in *optimeed.visualize.widgets.widget\_listWithSearch*), 127  
*Widget\_menuButton* (class in *optimeed.visualize*), 157  
*Widget\_menuButton* (class in *optimeed.visualize.widgets*), 131  
*Widget\_menuButton* (class in *optimeed.visualize.widgets.widget\_menuButton*), 128  
*Widget\_openGL* (class in *optimeed.visualize*), 157

*Widget\_openGL* (class in *optimeed.visualize.widgets*), 131  
*Widget\_openGL* (class in *optimeed.visualize.widgets.widget\_openGL*), 128  
*Widget\_tableWithSearch* (class in *optimeed.visualize*), 157  
*Widget\_tableWithSearch* (class in *optimeed.visualize.widgets*), 131  
*Widget\_tableWithSearch* (class in *optimeed.visualize.widgets.widget\_tableWithSearch*), 129  
*Widget\_text* (class in *optimeed.visualize*), 158  
*Widget\_text* (class in *optimeed.visualize.widgets*), 132  
*Widget\_text* (class in *optimeed.visualize.widgets.widget\_text*), 130  
*Widget\_text\_scrollable* (class in *optimeed.visualize*), 158  
*Widget\_text\_scrollable* (class in *optimeed.visualize.widgets*), 132  
*Widget\_text\_scrollable* (class in *optimeed.visualize.widgets.widget\_text*), 130  
*WindowHolders* (class in *optimeed.visualize*), 145  
*WindowHolders* (class in *optimeed.visualize.fastPlot*), 135

## Y

YELLOW (*text\_format* attribute), 47, 49  
*Yellow\_Emerald\_material* (in module *optimeed.visualize*), 155  
*Yellow\_Emerald\_material* (in module *optimeed.visualize.openGL*), 122  
*Yellow\_Emerald\_material* (in module *optimeed.visualize.openGL.materials*), 119