
optimeed

Release 2.0.2

Aug 07, 2022

Contents

1	Requirements	3
2	Installation	5
3	Quickstart	7
3.1	Quickstart Optimization	7
3.2	Quickstart Visualization	7
3.3	Loading and saving data	8
4	Gallery	11
4.1	Gallery	11
5	License and support	13
5.1	License and Support	13
6	API	15
6.1	<code>:py:mod:optimeed</code>	15
7	Developer guide	161
7.1	Developer documentation	161
	Python Module Index	163
	Index	165

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 ≤ 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.

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.gui.widgets`.

3.2.1 Simple gui using OpenGL:

3.2.2 Advanced visualization:

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

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:
```

(continues on next page)

(continued from previous page)

```
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 the object, read the file and interpret it 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`

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

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_optivariables ()

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 = "" ".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 ()
```

```
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 PrintIfShown

getCurrentShow ()

Get text type to be displayed by PrintIfShown

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 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 ()

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 OonclickInterface
    Interface class for the action to perform when a point is clicked
```

`onclick_animate`

Module Contents

Classes

```
class Oonclick_animate (theLinkDataGraph, theAnimation)
    Bases: optimeed.visualize.onclick.onclickInterface.OonclickInterface
    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

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_idenfifier*)

Action to perform when data stopped being selected :param selection_idenfifier: 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**Subpackages****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
```

`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)
        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 ()

```

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

Attributes

```
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)
        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 ()

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)

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

displayCollections

Module Contents

Classes

Functions

_is_object_selected (*object_in, min_max_attributes*)
_select_and_apply_action (*theCollections, min_max_attributes, theAction, selectionName*)
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 ()

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 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)

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_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 **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)
inkscape_svg_to_pdf (filename_svg, filename_pdf)
inkscape_svg_to_png (filename_svg, filename_png)
printIfShown (theStr, show_type=SHOW_DEBUG, isToPrint=True, appendTypeName=True, end='n')
SHOW_WARNING = 0
export_to_tikz_groupGraphs (theGraphs: optimeed.core.graphs.Graphs, foldername, additional-
                             Preamble=lambda: ", additionalAxisOptions=lambda graphId: ", ad-
                             ditionalTraceOptions=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

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*)

Overwritten to fix bad antialiasing while using `openGL`

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 if 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

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

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_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 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*)

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 ()

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)

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 **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 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 re-
turned 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 re-
        turned 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 re-
        turned by selector_updated :return:

    get_name ()
        Get the name of the action

        Returns string

```

6.1.2 Package Contents

VERSION = 2.1.0

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, [15](#)
optimeed consolidate, [15](#)
optimeed consolidate.fit, [15](#)
optimeed consolidate.parametric_analysis, [16](#)
optimeed consolidate.sensitivity_analysis, [17](#)
optimeed consolidate.sensitivity_analysis_evaluation, [19](#)
optimeed.core, [23](#)
optimeed.core.additional_tools, [26](#)
optimeed.core.ansi2html, [23](#)
optimeed.core.ansi2html.converter, [23](#)
optimeed.core.ansi2html.style, [25](#)
optimeed.core.ansi2html.util, [26](#)
optimeed.core.collection, [28](#)
optimeed.core.color_palette, [31](#)
optimeed.core.commonImport, [31](#)
optimeed.core.graphs, [32](#)
optimeed.core.graphs3, [36](#)
optimeed.core.inkscape_manager, [37](#)
optimeed.core.linkDataGraph, [37](#)
optimeed.core.myjson, [39](#)
optimeed.core.options, [40](#)
optimeed.core.tikzTranslator, [42](#)
optimeed.core.tools, [43](#)
optimeed.optimize, [62](#)
optimeed.optimize.characterization, [63](#)
optimeed.optimize.characterization.characterization, [63](#)
optimeed.optimize.characterization.interfaceCharacterization, [63](#)
optimeed.optimize.mathsToPhysics, [64](#)
optimeed.optimize.mathsToPhysics.interfaceMathsToPhysics, [64](#)
optimeed.optimize.mathsToPhysics.mathsToPhysics, [64](#)
optimeed.optimize.objAndCons, [65](#)
optimeed.optimize.objAndCons.fastObjCons, [65](#)
optimeed.optimize.objAndCons.interfaceObjCons, [66](#)
optimeed.optimize.optiAlgorithms, [66](#)
optimeed.optimize.optiAlgorithms.algorithmInterface, [72](#)
optimeed.optimize.optiAlgorithms.convergence, [66](#)
optimeed.optimize.optiAlgorithms.convergence.evolution, [66](#)
optimeed.optimize.optiAlgorithms.convergence.hyper, [67](#)
optimeed.optimize.optiAlgorithms.convergence.interface, [68](#)
optimeed.optimize.optiAlgorithms.monobjective_PSO, [72](#)
optimeed.optimize.optiAlgorithms.multiObjective_GA, [73](#)
optimeed.optimize.optiAlgorithms.NLOpt_Algorithm, [71](#)
optimeed.optimize.optiAlgorithms.pyswarm, [69](#)
optimeed.optimize.optiAlgorithms.pyswarm.pso, [69](#)
optimeed.optimize.optiHistoric, [77](#)
optimeed.optimize.optimizer, [80](#)
optimeed.optimize.optiVariable, [78](#)
optimeed.visualize, [86](#)
optimeed.visualize.displayCollections, [129](#)
optimeed.visualize.displayOptimization, [130](#)
optimeed.visualize.displaySensitivity, [130](#)
optimeed.visualize.fastPlot, [132](#)
optimeed.visualize.fastPlot3, [133](#)
optimeed.visualize.graphs, [86](#)
optimeed.visualize.graphs.colormap_pyqtgraph, [86](#)

optimeed.visualize.graphs.graphVisual, 86
 optimeed.visualize.graphs.pyqtgraphRedefined, 88
 optimeed.visualize.graphs.traceVisual, 91
 optimeed.visualize.graphs.widget_graphsViewport, 93
 optimeed.visualize.mainWindow, 133
 optimeed.visualize.onclick, 96
 optimeed.visualize.onclick.animation_example, 98
 optimeed.visualize.onclick.animationGUI, 96
 optimeed.visualize.onclick.collectionExportGUI, 99
 optimeed.visualize.onclick.onclick_animation, 100
 optimeed.visualize.onclick.onclick_changeStyle, 101
 optimeed.visualize.onclick.onclick_copyScreenshot, 101
 optimeed.visualize.onclick.onclick_delete, 102
 optimeed.visualize.onclick.onclick_exportCollection, 102
 optimeed.visualize.onclick.onclick_exportFile, 103
 optimeed.visualize.onclick.onclick_exportFrame, 103
 optimeed.visualize.onclick.onclick_extraFrame, 104
 optimeed.visualize.onclick.onclick_measure, 104
 optimeed.visualize.onclick.onclick_removeFrame, 105
 optimeed.visualize.onclick.onclick_representDevice, 105
 optimeed.visualize.onclick.onclick_tojson, 106
 optimeed.visualize.onclick.onclickInterface, 100
 optimeed.visualize.onclick.representDevice_examples, 106
 optimeed.visualize.process_mainloop, 133
 optimeed.visualize.selector, 114
 optimeed.visualize.selector.onselect_highlight, 114
 optimeed.visualize.selector.onselect_newTrace, 114
 optimeed.visualize.selector.onselect_splitTrace, 115
 optimeed.visualize.selector.onselectInterface, 114
 optimeed.visualize.viewOptimizationResults, 134
 optimeed.visualize.widgets, 117
 optimeed.visualize.widgets.openGL, 117
 optimeed.visualize.widgets.openGL.contextHandler, 117
 optimeed.visualize.widgets.openGL.deviceDrawerInterface, 118
 optimeed.visualize.widgets.openGL.materials, 118
 optimeed.visualize.widgets.openGL.openGL_library, 119
 optimeed.visualize.widgets.openGL.quaternions, 120
 optimeed.visualize.widgets.openGL.triangulate_polygon, 120
 optimeed.visualize.widgets.widget_doubleSlider, 122
 optimeed.visualize.widgets.widget_image, 122
 optimeed.visualize.widgets.widget_lineDrawer, 122
 optimeed.visualize.widgets.widget_listWithSearch, 123
 optimeed.visualize.widgets.widget_listWithSearchplus, 123
 optimeed.visualize.widgets.widget_menuButton, 124
 optimeed.visualize.widgets.widget_openGL, 124
 optimeed.visualize.widgets.widget_tableWithSearch, 124
 optimeed.visualize.widgets.widget_tableWithSearchplus, 125
 optimeed.visualize.widgets.widget_text, 126

Symbols

- `_AnimationTrace` (class in *optimeed.visualize.onclick.animationGUI*), 96
- `_AnimationTrace.AnimationElement` (class in *optimeed.visualize.onclick.animationGUI*), 96
- `_COMPRESS_SAVE_STR` (*ListDataStruct* attribute), 19, 29, 49
- `_COMPRESS_SAVE_STR` (*Performance_ListDataStruct* attribute), 30, 50
- `_DATA_STR` (*ListDataStruct* attribute), 19, 29, 49
- `_Device` (class in *optimeed consolidate.fit*), 15
- `_Evaluator` (class in *optimeed.optimize.optimizer*), 80
- `_LineItem` (class in *optimeed.visualize.onclick.onclick_measure*), 104
- `_NBR_ELEMENTS` (*Performance_ListDataStruct* attribute), 30, 50
- `_Objective` (class in *optimeed consolidate.fit*), 15
- `_OptiProjectLoader` (class in *optimeed.visualize.viewOptimizationResults*), 134
- `_PlotHolders` (class in *optimeed.visualize*), 142
- `_PlotHolders` (class in *optimeed.visualize.fastPlot*), 132
- `_STACK_SIZE` (*Performance_ListDataStruct* attribute), 30, 50
- `_State` (class in *optimeed.core.ansi2html.converter*), 24
- `__amb3__` (*MaterialRenderingProperties* attribute), 118, 121, 128, 151
- `__author__` (in module *optimeed.optimize.optiAlgorithms.convergence.hypervolume*), 67
- `__change_appearance_violate_constraints__` (*OptimizationDisplayer* method), 130, 136
- `__check_graphs__` (*Widget_graphsVisualLite* method), 94, 95, 148
- `__create_graph__` (*Widget_graphsVisualLite* method), 94, 95, 148
- `__dif3__` (*MaterialRenderingProperties* attribute), 118, 121, 128, 151
- `__draw_axis__` (*ContextHandler* method), 118
- `__getstate__` (*AutosaveStruct* method), 21, 29, 49
- `__len__` (*ListDataStruct* method), 19, 29, 49
- `__len__` (*MultiList* method), 68
- `__lightingInit__` (*ContextHandler* method), 118
- `__refresh__` (*OptimizationDisplayer* method), 130, 136
- `__reset__` (*ContextHandler* method), 118
- `__set_graphs_disposition__` (*OptimizationDisplayer* method), 130, 137
- `__setstate__` (*AutosaveStruct* method), 21, 29, 49
- `__shin__` (*MaterialRenderingProperties* attribute), 118, 121, 128, 151
- `__spec3__` (*MaterialRenderingProperties* attribute), 118, 121, 128, 151
- `__str__` (*AutosaveStruct* method), 20, 28, 49
- `__str__` (*Binary_OptimizationVariable* method), 79, 84
- `__str__` (*Data* method), 34, 54, 140
- `__str__` (*DataStruct_Interface* method), 28, 49
- `__str__` (*HowToPlotGraph* method), 37, 56
- `__str__` (*Integer_OptimizationVariable* method), 79, 84
- `__str__` (*InterfaceCharacterization* method), 63, 81
- `__str__` (*InterfaceObjCons* method), 66, 82
- `__str__` (*MathsToPhysics* method), 64, 65, 81
- `__str__` (*Monobjective_PSO* method), 73, 77, 83
- `__str__` (*MultiList* method), 68
- `__str__` (*MultiList.Node* method), 68
- `__str__` (*MultiObjective_GA* method), 75, 76, 83
- `__str__` (*NLOpt_Algorithm* method), 72
- `__str__` (*OptimizationVariable* method), 78
- `__str__` (*Option_class* method), 19, 41, 59
- `__str__` (*Real_OptimizationVariable* method), 79, 84
- `__str__` (*Rule* method), 25
- `__apply_regex__` (*Ansi2HTMLConverter* method), 25, 26

_apply_selector() (*CollectionDisplay* method), 129, 135
 _cancel_selector() (*CollectionDisplay* method), 129, 135
 _cellChanged() (*Widget_tableWithSearch* method), 125, 127, 151
 _collapse_cursor() (*Ansi2HTMLConverter* method), 25, 26
 _do_scatterPlot() (in module *optimeed.visualize*), 143
 _do_scatterPlot() (in module *optimeed.visualize.fastPlot3*), 133
 _extract_N_steps() (*EvolutionaryConvergence* method), 67, 69
 _filename_sensitivityparams (in module *optimeed consolidate.sensitivity_analysis*), 17
 _filename_sensitivityresults (in module *optimeed consolidate.sensitivity_analysis*), 17
 _filter_list() (*Widget_listWithSearch* method), 123, 127, 149
 _filter_list() (*Widget_tableWithSearch* method), 125, 128, 151
 _find_class() (in module *optimeed.core*), 61
 _find_class() (in module *optimeed.core.myjson*), 39
 _find_missings() (in module *optimeed consolidate.sensitivity_analysis*), 18
 _foldername_embarrassingly_parallel_results (in module *optimeed consolidate.sensitivity_analysis*), 17
 _format_data_lines() (*ListDataStruct* method), 20, 29, 50
 _format_fx_fs() (in module *optimeed.optimize.optiAlgorithms.pyswarm*), 70
 _format_fx_fs() (in module *optimeed.optimize.optiAlgorithms.pyswarm.pso*), 69
 _format_str_save() (*ListDataStruct* method), 20, 29, 50
 _get_S1_conv() (*SensitivityDisplay* method), 131, 136
 _get_ST_conv() (*SensitivityDisplay* method), 131, 136
 _get_annotations() (in module *optimeed.core*), 61
 _get_annotations() (in module *optimeed.core.myjson*), 40
 _get_attributes_to_save() (in module *optimeed.core*), 61
 _get_attributes_to_save() (in module *optimeed.core.myjson*), 40
 _get_job_args() (in module *optimeed consolidate.sensitivity_analysis*), 18
 _get_json_module_tree() (*ListDataStruct* method), 20, 29, 50
 _get_json_str_at_index() (*Performance_ListDataStruct* method), 30, 51
 _get_list_from_file() (*Performance_ListDataStruct* method), 30, 50
 _get_object_class() (in module *optimeed.core*), 61
 _get_object_class() (in module *optimeed.core.myjson*), 39
 _get_object_module() (in module *optimeed.core*), 61
 _get_object_module() (in module *optimeed.core.myjson*), 39
 _get_sensitivity_result() (in module *optimeed consolidate.sensitivity_analysis*), 18
 _get_sobol_indices() (*SensitivityDisplay* method), 131, 136
 _get_str_mainfile() (*Performance_ListDataStruct* method), 30, 50
 _html_template (in module *optimeed.core.ansi2html.converter*), 24
 _initialize() (*CollectionDisplay* method), 129, 135
 _initialize() (*Performance_ListDataStruct* method), 30, 50
 _instantiates_annotated_object() (in module *optimeed.core*), 61
 _instantiates_annotated_object() (in module *optimeed.core.myjson*), 40
 _is_feasible() (in module *optimeed.optimize.optiAlgorithms.pyswarm*), 70
 _is_feasible() (in module *optimeed.optimize.optiAlgorithms.pyswarm.pso*), 69
 _is_object_selected() (in module *optimeed.visualize.displayCollections*), 129
 _isclass() (in module *optimeed.core*), 61
 _isclass() (in module *optimeed.core.myjson*), 39
 _iter_items() (*Widget_listWithSearch* method), 123, 127, 149
 _iter_items() (*Widget_tableWithSearch* method), 125, 128, 151
 _latex_template (in module *optimeed.core.ansi2html.converter*), 24
 _map_index_to_file() (*Performance_ListDataStruct* method), 30, 51
 _needs_extra_newline() (in module *optimeed.core.ansi2html.converter*), 24
 _normalize_colors() (in module *optimeed.visualize.graphs.traceVisual*), 91
 _object_to_FQCN() (in module *optimeed.core*), 61
 _object_to_FQCN() (in module *optimeed.core.myjson*), 39
 _pack_options() (*Option_class* method), 19, 41, 59

`_remove_index_from_show()` (*_AnimationTrace method*), 97
`_remove_item_selector()` (*CollectionDisplay method*), 129, 135
`_reset_colors()` (*CollectionDisplay method*), 129, 135
`_save_module_tree()` (*Performance_ListDataStruct method*), 30, 51
`_select_and_apply_action()` (in module *optimeed.visualize.displayCollections*), 129
`_selector_to()` (*CollectionDisplay method*), 129, 135
`_set_study()` (*SensitivityDisplay method*), 131, 136
`_set_x()` (*CollectionDisplay method*), 129, 135
`_set_y()` (*CollectionDisplay method*), 129, 135
`_set_z()` (*CollectionDisplay method*), 129, 135
`_updateLabel()` (*myAxis method*), 90
`_workspace_path` (in module *optimeed.core*), 45
`_workspace_path` (in module *optimeed.core.tools*), 43

A

`activateMaterialProperties()` (*MaterialRenderingProperties method*), 119, 121, 128, 151
`add()` (*SeveralTerminationCondition method*), 74
`add_action_on_click()` (in module *optimeed.visualize*), 142
`add_action_on_click()` (in module *optimeed.visualize.fastPlot*), 132
`add_action_on_click()` (*WindowHolders method*), 132, 142
`add_collection()` (*CollectionDisplay method*), 129, 135
`add_collection()` (*LinkDataGraph method*), 37, 56
`add_data()` (*GraphVisual method*), 87, 147
`add_data()` (*ListDataStruct method*), 20, 29, 50
`add_data()` (*Performance_ListDataStruct method*), 30, 50
`add_data()` (*SensitivityResults method*), 17
`add_data_to_collection()` (*CollectionExporter-GUI method*), 100
`add_element()` (*_AnimationTrace method*), 97
`add_elementToTrace()` (*AnimationGUI method*), 97
`add_feature()` (*GraphVisual method*), 87, 147
`add_graph()` (*Graphs method*), 35, 55, 141
`add_graph()` (*LinkDataGraph method*), 38, 57
`add_index_to_show()` (*_AnimationTrace method*), 97
`add_json_data()` (*Performance_ListDataStruct method*), 30, 51
`add_modified_paintElem()` (*TraceVisual._ModifiedPaintElem method*), 91, 144
`add_opti_project()` (*ViewOptimizationResults method*), 134, 137
`add_option()` (*Option_class method*), 19, 41, 59
`add_parameters()` (*OptiHistoric._LogParams method*), 78, 85
`add_plot()` (*_PlotHolders method*), 132, 142
`add_plot()` (*WindowHolders method*), 132, 142
`add_point()` (*ConvergenceManager method*), 71
`add_point()` (*Data method*), 34, 53, 140
`add_prefix_attribute_name()` (*Optimization-Variable method*), 78
`add_study()` (*SensitivityDisplay method*), 131, 135
`add_suffix_to_path()` (in module *optimeed.core*), 46
`add_suffix_to_path()` (in module *optimeed.core.tools*), 44
`add_terminationCondition()` (*MultiObjective_GA method*), 75, 76, 83
`add_trace()` (*AnimationGUI method*), 97
`add_trace()` (*Graph method*), 34, 54
`add_trace()` (*Graphs method*), 35, 54, 140
`add_trace()` (*GraphVisual method*), 87, 147
`add_trace_firstGraph()` (*Graphs method*), 35, 54, 140
`add_update_method()` (*Graphs method*), 35, 55, 141
`add_visual()` (*OnClick_representDevice.DataInformationVisuals method*), 105, 111, 152
`addItem()` (*myGraphicsLayout method*), 89
`addItem()` (*myLegend method*), 90
`adjust()` (*_State method*), 24
ALGORITHM (*NLOpt_Algorithm attribute*), 71
AlgorithmInterface (class in *optimeed.optimize.optiAlgorithms.algorithmInterface*), 72
`analyse_sobol_convergence()` (in module *optimeed consolidate*), 22
`analyse_sobol_convergence()` (in module *optimeed consolidate.sensitivity_analysis*), 18
`analyse_sobol_create_array()` (in module *optimeed consolidate.sensitivity_analysis*), 18
`analyse_sobol_plot_2ndOrder_indices()` (in module *optimeed.visualize*), 136
`analyse_sobol_plot_2ndOrder_indices()` (in module *optimeed.visualize.displaySensitivity*), 131
`analyse_sobol_plot_convergence()` (in module *optimeed.visualize*), 136
`analyse_sobol_plot_convergence()` (in module *optimeed.visualize.displaySensitivity*), 131
`analyse_sobol_plot_indices()` (in module *optimeed.visualize*), 136
`analyse_sobol_plot_indices()` (in module *optimeed.visualize.displaySensitivity*), 131

- `Animate_lines` (class in *optimeed.visualize*), 157
- `Animate_lines` (class in *optimeed.visualize.onclick*), 112
- `Animate_lines` (class in *optimeed.visualize.onclick.animation_examples*), 99
- `Animate_lines_and_text` (class in *optimeed.visualize*), 157
- `Animate_lines_and_text` (class in *optimeed.visualize.onclick*), 113
- `Animate_lines_and_text` (class in *optimeed.visualize.onclick.animation_examples*), 99
- `Animate_openGL` (class in *optimeed.visualize*), 157
- `Animate_openGL` (class in *optimeed.visualize.onclick*), 113
- `Animate_openGL` (class in *optimeed.visualize.onclick.animation_examples*), 98
- `Animate_openGL_and_text` (class in *optimeed.visualize*), 158
- `Animate_openGL_and_text` (class in *optimeed.visualize.onclick*), 113
- `Animate_openGL_and_text` (class in *optimeed.visualize.onclick.animation_examples*), 98
- `AnimationGUI` (class in *optimeed.visualize.onclick.animationGUI*), 97
- `Ansi2HTMLConverter` (class in *optimeed.core.ansi2html*), 26
- `Ansi2HTMLConverter` (class in *optimeed.core.ansi2html.converter*), 25
- `ANSI_BACKGROUND_256` (in module *optimeed.core.ansi2html.converter*), 24
- `ANSI_BACKGROUND_CUSTOM_MAX` (in module *optimeed.core.ansi2html.converter*), 24
- `ANSI_BACKGROUND_CUSTOM_MIN` (in module *optimeed.core.ansi2html.converter*), 24
- `ANSI_BACKGROUND_DEFAULT` (in module *optimeed.core.ansi2html.converter*), 24
- `ANSI_BACKGROUND_HIGH_INTENSITY_MAX` (in module *optimeed.core.ansi2html.converter*), 24
- `ANSI_BACKGROUND_HIGH_INTENSITY_MIN` (in module *optimeed.core.ansi2html.converter*), 24
- `ANSI_BLINK_FAST` (in module *optimeed.core.ansi2html.converter*), 23
- `ANSI_BLINK_OFF` (in module *optimeed.core.ansi2html.converter*), 23
- `ANSI_BLINK_SLOW` (in module *optimeed.core.ansi2html.converter*), 23
- `ANSI_CROSSED_OUT_OFF` (in module *optimeed.core.ansi2html.converter*), 23
- `ANSI_CROSSED_OUT_ON` (in module *optimeed.core.ansi2html.converter*), 23
- `ANSI_FOREGROUND_256` (in module *optimeed.core.ansi2html.converter*), 24
- `ANSI_FOREGROUND_CUSTOM_MAX` (in module *optimeed.core.ansi2html.converter*), 24
- `ANSI_FOREGROUND_CUSTOM_MIN` (in module *optimeed.core.ansi2html.converter*), 24
- `ANSI_FOREGROUND_DEFAULT` (in module *optimeed.core.ansi2html.converter*), 24
- `ANSI_FOREGROUND_HIGH_INTENSITY_MAX` (in module *optimeed.core.ansi2html.converter*), 24
- `ANSI_FOREGROUND_HIGH_INTENSITY_MIN` (in module *optimeed.core.ansi2html.converter*), 24
- `ANSI_FULL_RESET` (in module *optimeed.core.ansi2html.converter*), 23
- `ANSI_INTENSITY_INCREASED` (in module *optimeed.core.ansi2html.converter*), 23
- `ANSI_INTENSITY_NORMAL` (in module *optimeed.core.ansi2html.converter*), 23
- `ANSI_INTENSITY_REDUCED` (in module *optimeed.core.ansi2html.converter*), 23
- `ANSI_NEGATIVE_OFF` (in module *optimeed.core.ansi2html.converter*), 24
- `ANSI_NEGATIVE_ON` (in module *optimeed.core.ansi2html.converter*), 24
- `ANSI_STYLE_ITALIC` (in module *optimeed.core.ansi2html.converter*), 23
- `ANSI_STYLE_NORMAL` (in module *optimeed.core.ansi2html.converter*), 23
- `ANSI_UNDERLINE_OFF` (in module *optimeed.core.ansi2html.converter*), 23
- `ANSI_UNDERLINE_ON` (in module *optimeed.core.ansi2html.converter*), 23
- `ANSI_VISIBILITY_OFF` (in module *optimeed.core.ansi2html.converter*), 23
- `ANSI_VISIBILITY_ON` (in module *optimeed.core.ansi2html.converter*), 23
- `app` (in module *optimeed.visualize.process_mainloop*), 133
- `append()` (*MultiList* method), 68
- `apply_module_tree_to_dict()` (in module *optimeed.core*), 48, 62
- `apply_module_tree_to_dict()` (in module *optimeed.core.myjson*), 40
- `apply_palette()` (*GraphVisual* method), 87, 147
- `apply_regex()` (*Ansi2HTMLConverter* method), 25, 26
- `apply_width_sample()` (*myLegend* method), 90
- `applyEquation()` (in module *optimeed.core*), 45
- `applyEquation()` (in module *optimeed.core.tools*), 43
- `arithmeticEval()` (in module *optimeed.core*), 46
- `arithmeticEval()` (in module *optimeed.core.tools*), 43
- `attributeName` (*OptimizationVariable* attribute), 78

`attrs()` (*Ansi2HTMLConverter method*), 25, 26
`AutosaveStruct` (*class in optimeed consolidate*), 20
`AutosaveStruct` (*class in optimeed core*), 49
`AutosaveStruct` (*class in optimeed core collection*), 28
`axis_equal()` (*_PlotHolders method*), 132, 142
`axis_equal()` (*GraphVisual method*), 88, 147
`axis_equal()` (*in module optimeed visualize*), 142
`axis_equal()` (*in module optimeed visualize fastPlot*), 133
`axis_equal()` (*WindowHolders method*), 132, 142
`axisangle_to_q()` (*in module optimeed visualize widgets.openGL.quaternions*), 120

B

`Base_Option` (*class in optimeed core*), 58
`Base_Option` (*class in optimeed core options*), 40
`Binary_OptimizationVariable` (*class in optimeed optimize*), 84
`Binary_OptimizationVariable` (*class in optimeed optimize optiVariable*), 79
`blackOnly()` (*in module optimeed core*), 51
`blackOnly()` (*in module optimeed core color_palette*), 31
`BLUE` (*text_format attribute*), 43, 45
`Blue_material` (*in module optimeed visualize*), 152
`Blue_material` (*in module optimeed visualize widgets*), 128
`Blue_material` (*in module optimeed visualize widgets.openGL*), 121
`Blue_material` (*in module optimeed visualize widgets.openGL.materials*), 119
`BOLD` (*text_format attribute*), 43, 45
`boundingRect()` (*_LineItem method*), 104
`Brass_material` (*in module optimeed visualize*), 151
`Brass_material` (*in module optimeed visualize widgets*), 128
`Brass_material` (*in module optimeed visualize widgets.openGL*), 121
`Brass_material` (*in module optimeed visualize widgets.openGL.materials*), 119
`Bronze_material` (*in module optimeed visualize*), 151
`Bronze_material` (*in module optimeed visualize widgets*), 128
`Bronze_material` (*in module optimeed visualize widgets.openGL*), 121
`Bronze_material` (*in module optimeed visualize widgets.openGL.materials*), 119

C

`cancel_selector()` (*Onselect_highlight method*), 114, 116, 158
`cancel_selector()` (*Onselect_newTrace method*), 115, 116, 159
`cancel_selector()` (*Onselect_splitTrace method*), 115, 117, 159
`cart2pol()` (*in module optimeed core*), 59
`cart2pol()` (*in module optimeed core additional_tools*), 27
`cellChanged` (*Widget_tableWithSearch attribute*), 125, 127, 150
`Characterization` (*class in optimeed optimize*), 81
`Characterization` (*class in optimeed optimize characterization*), 63
`Characterization` (*class in optimeed optimize characterization characterization*), 63
`check_if_must_plot()` (*in module optimeed visualize displayOptimization*), 130
`Chrome_material` (*in module optimeed visualize*), 152
`Chrome_material` (*in module optimeed visualize widgets*), 128
`Chrome_material` (*in module optimeed visualize widgets.openGL*), 121
`Chrome_material` (*in module optimeed visualize widgets.openGL.materials*), 119
`CLASS_TAG` (*in module optimeed core*), 61
`CLASS_TAG` (*in module optimeed core myjson*), 39
`CLIC_LEFT` (*in module optimeed visualize widgets.openGL.contextHandler*), 117
`CLIC_RIGHT` (*in module optimeed visualize widgets.openGL.contextHandler*), 117
`clone()` (*ListDataStruct method*), 20, 29, 49
`clone()` (*Performance_ListDataStruct method*), 30, 50
`close()` (*MyMultiprocessEvaluator method*), 75
`close_windows()` (*OptimizationDisplayer method*), 130, 137
`closeEvent()` (*AnimationGUI method*), 98
`closeEvent()` (*MainWindow method*), 133, 138
`cmapToColormap()` (*in module optimeed visualize graphs.colormap_pyqtgraph*), 86
`CollectionDisplayer` (*class in optimeed visualize*), 135
`CollectionDisplayer` (*class in optimeed visualize displayCollections*), 129
`CollectionExporterGUI` (*class in optimeed visualize onclick.collectionExporterGUI*), 99

`color()` (in module `optimeed.core.ansi2html.style`), 25
`color_component()` (in module `optimeed.core.ansi2html.style`), 25
`compute()` (*_Objective method*), 15
`compute()` (*Characterization method*), 63, 81
`compute()` (*FastObjCons method*), 65, 66, 82
`compute()` (*HyperVolume method*), 67
`compute()` (*Monobjective_PSO method*), 72, 77, 83
`compute()` (*MultiObjective_GA method*), 75, 76, 82
`compute()` (*NLOpt_Algorithm method*), 71
`constraints` (*OptiHistoric._pointData attribute*), 77, 85
`constraints_per_step` (*EvolutionaryConvergence attribute*), 67, 69
`contains_trace()` (*AnimationGUI method*), 98
`ContextHandler` (class in `optimeed.visualize.widgets.openGL.contextHandler`), 117
`ContourPlot` (class in `optimeed.core`), 56
`ContourPlot` (class in `optimeed.core.graphs3`), 36
`ContourPlot` (class in `optimeed.visualize`), 143
`ConvergenceManager` (class in `optimeed.optimize.optiAlgorithms.NLOpt_Algorithm`), 71
`ConvergenceTerminationCondition` (class in `optimeed.optimize.optiAlgorithms.multiObjective_GA`), 74
`convert()` (*Ansi2HTMLConverter method*), 25, 26
`convert_color()` (in module `optimeed.core`), 60
`convert_color()` (in module `optimeed.core.additional_tools`), 28
`convert_color_with_alpha()` (in module `optimeed.core`), 51, 60
`convert_color_with_alpha()` (in module `optimeed.core.additional_tools`), 28
`convert_linestyle()` (in module `optimeed.core.tikzTranslator`), 42
`convert_marker()` (in module `optimeed.core.tikzTranslator`), 42
`convert_to_gridplot()` (in module `optimeed.core`), 56
`convert_to_gridplot()` (in module `optimeed.core.graphs3`), 36
`Copper_material` (in module `optimeed.visualize`), 152
`Copper_material` (in module `optimeed.visualize.widgets`), 128
`Copper_material` (in module `optimeed.visualize.widgets.openGL`), 121
`Copper_material` (in module `optimeed.visualize.widgets.openGL.materials`), 119
`create_main_window()` (*OptimizationDisplayer method*), 130, 137
`create_unique_dirname()` (in module `optimeed.core`), 45
`create_unique_dirname()` (in module `optimeed.core.tools`), 43
`createWidget()` (*Plugin_listWithSearch method*), 123
`createWidget()` (*Plugin_tableWithSearch method*), 125
`curr_index()` (*OnClick_representDevice.DataInformationVisuals method*), 105, 111, 152
`CursorMoveUp` (class in `optimeed.core.ansi2html.converter`), 24
`CYAN` (*text_format attribute*), 43, 45
`Cyan_material` (in module `optimeed.visualize`), 152
`Cyan_material` (in module `optimeed.visualize.widgets`), 129
`Cyan_material` (in module `optimeed.visualize.widgets.openGL`), 121
`Cyan_material` (in module `optimeed.visualize.widgets.openGL.materials`), 119
D
`dark2()` (in module `optimeed.core`), 51
`dark2()` (in module `optimeed.core.color_palette`), 31
`DARKCYAN` (*text_format attribute*), 43, 45
`Data` (class in `optimeed.core`), 51
`Data` (class in `optimeed.core.graphs`), 32
`Data` (class in `optimeed.visualize`), 138
`DataStruct_Interface` (class in `optimeed.core`), 49
`DataStruct_Interface` (class in `optimeed.core.collection`), 28
`decode_str_json()` (in module `optimeed.core`), 48, 62
`decode_str_json()` (in module `optimeed.core.myjson`), 40
`deep_sizeof()` (in module `optimeed.core`), 47
`deep_sizeof()` (in module `optimeed.core.tools`), 45
`default` (in module `optimeed.optimize.optimizer`), 80
`default_colormap` (in module `optimeed.visualize.graphs.traceVisual`), 91
`default_palette()` (in module `optimeed.core`), 51
`default_palette()` (in module `optimeed.core.color_palette`), 31
`delete()` (*GraphVisual method*), 88, 147
`delete()` (*Widget_graphsVisualLite method*), 94, 96, 149
`delete_all()` (*_AnimationTrace method*), 97
`delete_all()` (*AnimationGUI method*), 97
`delete_clicked_item()` (*LinkDataGraph method*), 39, 58

delete_clicked_items() (*LinkDataGraph method*), 39, 58
 delete_graph() (*Widget_graphsVisualLite method*), 94, 96, 149
 delete_indices_from_list() (*in module optimeed.core*), 47, 48
 delete_indices_from_list() (*in module optimeed.core.tools*), 44
 delete_key_widgets() (*Animate_lines method*), 99, 112, 157
 delete_key_widgets() (*Animate_openGL method*), 98, 113, 157
 delete_lines() (*Widget_lineDrawer method*), 122, 126, 150
 delete_point() (*AnimationGUI method*), 97
 delete_point() (*Data method*), 34, 53, 140
 delete_points_at_indices() (*ListDataStruct method*), 20, 29, 50
 delete_points_at_indices() (*Performance_ListDataStruct method*), 31, 51
 delete_trace() (*GraphVisual method*), 88, 147
 delete_visual() (*OnClick_representDevice.DataInformationVisual method*), 105, 111, 152
 derivate() (*in module optimeed.core*), 59
 derivate() (*in module optimeed.core.additional_tools*), 27
 DeviceDrawerInterface (class *in optimeed.visualize*), 151
 DeviceDrawerInterface (class *in optimeed.visualize.widgets*), 128
 DeviceDrawerInterface (class *in optimeed.visualize.widgets.openGL*), 121
 DeviceDrawerInterface (class *in optimeed.visualize.widgets.openGL.deviceDrawerInterface*), 118
 disableLogs() (*in module optimeed.core*), 47
 disableLogs() (*in module optimeed.core.commonImport*), 31
 display_distance() (*OnClick_measure method*), 105, 110, 142, 155
 display_graphs() (*OptimizationDisplayer method*), 130, 137
 display_graphs() (*ViewOptimizationResults method*), 134, 137
 dist() (*in module optimeed.core*), 60
 dist() (*in module optimeed.core.additional_tools*), 27
 DIVISION_OUTER (*MultiObjective_GA attribute*), 75, 76, 82
 do_fit() (*in module optimeed consolidate*), 22
 do_fit() (*in module optimeed consolidate.fit*), 16
 do_generate_figure() (*in module optimeed.core.tikzTranslator*), 42
 do_MathsToPhys() (*Binary_OptimizationVariable method*), 79, 84
 do_MathsToPhys() (*Integer_OptimizationVariable method*), 79, 84
 do_MathsToPhys() (*OptimizationVariable method*), 78
 do_MathsToPhys() (*Real_OptimizationVariable method*), 79, 84
 do_preamble() (*in module optimeed.core.tikzTranslator*), 42
 do_preamble3D() (*in module optimeed.core.tikzTranslator*), 42
 do_specific_axis_options() (*in module optimeed.core.tikzTranslator*), 42
 do_specific_trace_options() (*in module optimeed.core.tikzTranslator*), 42
 doubleValueChanged (widget_doubleSlider attribute), 122
 draw_2Dring() (*in module optimeed.visualize.widgets.openGL.openGL_library*), 119
 draw_2Dring_diff_angle() (*in module optimeed.visualize.widgets.openGL.openGL_library*), 119
 draw_carved_disk() (*in module optimeed.visualize.widgets.openGL.openGL_library*), 120
 draw_closedPolygon() (*in module optimeed.visualize.widgets.openGL.openGL_library*), 119
 draw_cylinder() (*in module optimeed.visualize.widgets.openGL.openGL_library*), 119
 draw_disk() (*in module optimeed.visualize.widgets.openGL.openGL_library*), 120
 draw_extrudeZ() (*in module optimeed.visualize.widgets.openGL.openGL_library*), 119
 draw_lines() (*in module optimeed.visualize.widgets.openGL.openGL_library*), 119
 draw_part_cylinder() (*in module optimeed.visualize.widgets.openGL.openGL_library*), 120
 draw_part_cylinder_throat() (*in module optimeed.visualize.widgets.openGL.openGL_library*), 120
 draw_part_disk() (*in module optimeed.visualize.widgets.openGL.openGL_library*), 120
 draw_part_disk_diff_angles() (*in module optimeed.visualize.widgets.openGL.openGL_library*), 120
 draw_rectangle() (*in module optimeed.visualize.widgets.openGL.openGL_library*), 120

meed.visualize.widgets.openGL.openGL_library), *evaluate_all()* (*MyMapEvaluator* method), 69, 70, 119
draw_simple_rectangle() (in module *optimeed.visualize.widgets.openGL.openGL_library*), 119
draw_spiral() (in module *optimeed.visualize.widgets.openGL.openGL_library*), 119
draw_spiralFront() (in module *optimeed.visualize.widgets.openGL.openGL_library*), 119
draw_spiralFull() (in module *optimeed.visualize.widgets.openGL.openGL_library*), 119
draw_spiralSheet() (in module *optimeed.visualize.widgets.openGL.openGL_library*), 119
draw_triList() (in module *optimeed.visualize.widgets.openGL.openGL_library*), 119
draw_tubeSheet() (in module *optimeed.visualize.widgets.openGL.openGL_library*), 119
drawWireTube() (in module *optimeed.visualize.widgets.openGL.openGL_library*), 120
E
Emerald_material (in module *optimeed.visualize*), 151
Emerald_material (in module *optimeed.visualize.widgets*), 128
Emerald_material (in module *optimeed.visualize.widgets.openGL*), 121
Emerald_material (in module *optimeed.visualize.widgets.openGL.materials*), 119
emitDoubleValueChanged() (*widget_doubleSlider* method), 122
enableLogs() (in module *optimeed.core*), 47
enableLogs() (in module *optimeed.core.commonImport*), 31
encode_str_json() (in module *optimeed.core*), 48, 62
encode_str_json() (in module *optimeed.core.myjson*), 40
END (*text_format* attribute), 43, 45
evaluate() (*_Evaluator* method), 80
evaluate() (in module *optimeed consolidate.sensitivity_analysis_evaluation*), 19
evaluate() (*MyProblem* method), 73
evaluate() (*Parametric_analysis* method), 17, 21
evaluate_all() (*MyMultiprocessEvaluator* method), 70, 75
evaluate_sensitivities() (in module *optimeed consolidate*), 22
evaluate_sensitivities() (in module *optimeed consolidate.sensitivity_analysis*), 18
eventFilter() (*Widget_image* method), 122, 126, 149
EvolutionaryConvergence (class in *optimeed.optimize.optiAlgorithms.convergence*), 69
EvolutionaryConvergence (class in *optimeed.optimize.optiAlgorithms.convergence.evolutionaryConvergence*), 67
EXCLUDED_TAGS (in module *optimeed.core*), 61
EXCLUDED_TAGS (in module *optimeed.core.myjson*), 39
export_picture() (*AnimationGUI* method), 98
export_str() (*Data* method), 34, 54, 140
export_str() (*Graph* method), 34, 54
export_str() (*Graphs* method), 35, 55, 141
export_svg() (*Widget_graphsVisualLite* method), 94, 95, 148
export_tikz() (*Widget_graphsVisualLite* method), 94, 96, 148
export_to_tikz_contour_plot() (in module *optimeed.core*), 62
export_to_tikz_contour_plot() (in module *optimeed.core.tikzTranslator*), 42
export_to_tikz_groupGraphs() (in module *optimeed.core*), 62
export_to_tikz_groupGraphs() (in module *optimeed.core.tikzTranslator*), 42
export_to_tikz_groupGraphs() (in module *optimeed.visualize*), 143
export_txt() (*Widget_graphsVisualLite* method), 94, 95, 148
export_widget() (*Animate_lines* method), 99, 112, 157
export_widget() (*Animate_openGL* method), 98, 113, 157
exportCollection() (*CollectionExporterGUI* method), 100
exportGraphs() (*Widget_graphsVisualLite* method), 94, 95, 148
extend() (*MultiList* method), 68
extract_collection_from_indices() (*ListDataStruct* method), 20, 29, 49
extract_collection_from_indices() (*Performance_ListDataStruct* method), 30, 50
F
fast_LUT_interpolation (class in *opti-*

meed.core), 59
 fast_LUT_interpolation (class in *optimeed.core.additional_tools*), 26
 fast_update() (*GraphVisual* method), 88, 147
 fast_update() (*Widget_graphsVisualLite* method), 94, 95, 148
 FastObjCons (class in *optimeed.optimize*), 82
 FastObjCons (class in *optimeed.optimize.objAndCons*), 66
 FastObjCons (class in *optimeed.optimize.objAndCons.fastObjCons*), 65
 figure() (in module *optimeed.visualize*), 142
 figure() (in module *optimeed.visualize.fastPlot*), 132
 FilledContourPlot (class in *optimeed.core*), 56
 FilledContourPlot (class in *optimeed.core.graphs3*), 36
 FilledContourPlot (class in *optimeed.visualize*), 143
 find_all_colors() (in module *optimeed.core.tikzTranslator*), 42
 find_and_replace() (in module *optimeed.core*), 45
 find_and_replace() (in module *optimeed.core.tools*), 43
 force_hide_row() (*Widget_tableWithSearch* method), 125, 127, 150
 format_escape_char() (in module *optimeed.core.tikzTranslator*), 42
 format_Griddata() (in module *optimeed.core.tikzTranslator*), 42
 format_scatterdata() (in module *optimeed.core.tikzTranslator*), 42
 frame_selector() (*AnimationGUI* method), 97
 fromMathsToPhys() (*MathsToPhysics* method), 64, 65, 81
 fromPhysToMaths() (*MathsToPhysics* method), 64, 65, 81

G

gather_embarrassingly_parallel_sensitivity() (in module *optimeed consolidate*), 23
 gather_embarrassingly_parallel_sensitivity() (in module *optimeed consolidate.sensitivity_analysis*), 18
 generate() (*MyGenerator* method), 73
 generate_optimizationGraphs() (*OptimizationDisplayer* method), 130, 136
 get() (*_AnimationTrace.AnimationElement* method), 97
 get_2D_pareto() (in module *optimeed.core*), 46
 get_2D_pareto() (in module *optimeed.core.tools*), 44
 get_additional_attributes_to_save() (*OptimizerSettings* method), 80, 85
 get_additional_attributes_to_save() (*SaveableObject* method), 39, 61
 get_additional_attributes_to_save() (*SensitivityParameters* method), 18, 22
 get_additional_attributes_to_save() (*SensitivityResults* method), 17
 get_additional_attributes_to_save_list() (*OptimizerSettings* method), 80, 85
 get_additional_attributes_to_save_list() (*SaveableObject* method), 39, 61
 get_all_figures() (in module *optimeed.visualize*), 143
 get_all_figures() (in module *optimeed.visualize.fastPlot*), 133
 get_all_figures() (*WindowHolders* method), 132, 142
 get_all_graphs() (*Graphs* method), 35, 55, 141
 get_all_graphs_ids() (*Graphs* method), 35, 55, 141
 get_all_graphsVisual() (*Widget_graphsVisualLite* method), 94, 96, 149
 get_all_traces() (*Graph* method), 34, 54
 get_all_traces() (*GraphVisual* method), 88, 147
 get_all_traces_ids() (*Graph* method), 34, 54
 get_alpha() (*Data* method), 32, 52, 139
 get_analyzed_attribute() (*Parametric_parameter* method), 16, 21
 get_attribute_name() (*OptimizationVariable* method), 78
 get_axis() (*GraphVisual* method), 87, 146
 get_base_pen() (*_AnimationTrace* method), 97
 get_base_pen() (*TraceVisual* method), 91, 144
 get_base_symbol() (*TraceVisual* method), 92, 144
 get_base_symbol_brush() (*TraceVisual* method), 91, 144
 get_base_symbol_pen() (*TraceVisual* method), 91, 144
 get_best_devices_without_reevaluating() (*OptiHistoric* method), 78, 85
 get_brushes() (*TraceVisual* method), 92, 145
 get_charac() (*OptimizerSettings* method), 80, 85
 get_charac() (*SensitivityParameters* method), 18, 22
 get_choices() (*Base_Option* method), 40, 58
 get_choices() (*Option_bool* method), 41, 58
 get_clicked_item() (*LinkDataGraph* method), 38, 57
 get_clicked_items() (*LinkDataGraph* method), 38, 58
 get_collection() (*LinkDataGraph* method), 38, 57
 get_collection_from_graph() (*LinkDataGraph* method), 38, 57
 get_color() (*Data* method), 32, 52, 138
 get_color() (*ScatterPlot3* method), 36, 56, 143
 get_color() (*TraceVisual* method), 91, 144

`get_color_alpha()` (*Data method*), 32, 52, 139
`get_colour_background()` (*DeviceDrawerInterface method*), 118, 121, 128, 151
`get_colour_scalebar()` (*DeviceDrawerInterface method*), 118, 121, 128, 151
`get_constraints()` (*OptimizerSettings method*), 80, 85
`get_convergence()` (*_OptiProjectLoader method*), 134
`get_convergence()` (*Monobjective_PSO method*), 73, 77, 83
`get_convergence()` (*MultiObjective_GA method*), 75, 76, 83
`get_convergence()` (*NLOpt_Algorithm method*), 72
`get_convergence()` (*OptiHistoric method*), 78, 86
`get_curr_plotHolder()` (*WindowHolders method*), 132, 142
`get_data()` (*ListDataStruct method*), 20, 29, 50
`get_data()` (*TraceVisual method*), 92, 145
`get_data_at_index()` (*ListDataStruct method*), 20, 29, 50
`get_data_at_index()` (*Performance_ListDataStruct method*), 30, 51
`get_data_generator()` (*ListDataStruct method*), 20, 29, 50
`get_data_generator()` (*Performance_ListDataStruct method*), 30, 51
`get_data_link()` (*ViewOptimizationResults method*), 134, 137
`get_dataIndex_from_graphIndex()` (*Data method*), 33, 53, 139
`get_dataIndices_from_graphIndices()` (*Data method*), 33, 53, 139
`get_datalink()` (*CollectionDisplay method*), 129, 135
`get_datastruct()` (*AutosaveStruct method*), 21, 29, 49
`get_device()` (*OptimizerSettings method*), 80, 85
`get_device()` (*SensitivityParameters method*), 17, 22
`get_devices()` (*_OptiProjectLoader method*), 134
`get_devices()` (*OptiHistoric method*), 78, 86
`get_element_animations()` (*_AnimationTrace method*), 97
`get_ellipse_axes()` (*in module optimeed.core*), 60
`get_ellipse_axes()` (*in module optimeed.core.additional_tools*), 27
`get_entries_selected()` (*Widget_get_tableWithSearch method*), 125, 127, 150
`get_extrema_lines()` (*Widget_lineDrawer method*), 123, 126, 150
`get_filename()` (*AutosaveStruct method*), 20, 28, 49
`get_first_graph()` (*Graphs method*), 35, 55, 141
`get_graph()` (*Graphs method*), 35, 55, 141
`get_graph()` (*Widget_graphsVisualLite method*), 94, 96, 148
`get_graph_and_trace_from_idCollection()` (*LinkDataGraph method*), 39, 58
`get_graphIndex_from_dataIndex()` (*Data method*), 33, 53, 139
`get_graphIndices_from_dataIndices()` (*Data method*), 33, 53, 139
`get_graphs()` (*EvolutionaryConvergence method*), 67, 69
`get_graphs()` (*LinkDataGraph method*), 38, 57
`get_howToPlotGraph()` (*LinkDataGraph method*), 38, 57
`get_hypervolume()` (*EvolutionaryConvergence method*), 67, 69
`get_hypervolume_convergence()` (*EvolutionaryConvergence method*), 67, 69
`get_idcollection_from_collection()` (*LinkDataGraph method*), 39, 58
`get_idCollection_from_graph()` (*LinkDataGraph method*), 38, 57
`get_idCollections()` (*LinkDataGraph method*), 38, 57
`get_idGraphs()` (*LinkDataGraph method*), 38, 57
`get_idPoints_from_indices_in_collection()` (*LinkDataGraph method*), 39, 58
`get_idTraces()` (*LinkDataGraph method*), 38, 57
`get_index_selected()` (*Widget_listWithSearch method*), 123, 126, 149
`get_indices_points_to_plot()` (*Data method*), 34, 54, 140
`get_indices_to_show()` (*_AnimationTrace method*), 97
`get_inkscape_version()` (*in module optimeed.core*), 62
`get_inkscape_version()` (*in module optimeed.core.inkscape_manager*), 37
`get_interesting_elements()` (*Animate_lines method*), 99, 112, 157
`get_interesting_elements()` (*Animate_openGL_and_text method*), 98, 113, 158
`get_invert_permutations()` (*Data method*), 33, 53, 139
`get_item()` (*Widget_tableWithSearch method*), 125, 127, 151
`get_json_module_tree_from_dict()` (*in module optimeed.core*), 48, 61
`get_json_module_tree_from_dict()` (*in module optimeed.core.myjson*), 40
`get_kwargs()` (*_OptiProjectLoader method*), 134
`get_label()` (*Plot3D_Generic method*), 36, 56
`get_label_pos()` (*myAxis method*), 90
`get_layout_buttons()` (*Widget_graphsVisualLite*

- method*), 94, 96, 149
- `get_legend()` (*Data method*), 33, 53, 140
- `get_legend()` (*GraphVisual method*), 87, 146
- `get_legend()` (*Plot3D_Generic method*), 36, 56
- `get_length()` (*ListDataStruct method*), 20, 29, 49
- `get_length()` (*TraceVisual method*), 92, 145
- `get_length_data()` (*Data method*), 32, 52, 138
- `get_levels()` (*ContourPlot method*), 36, 56, 143
- `get_lim()` (*Plot3D_Generic method*), 36, 56
- `get_linestyle()` (*Data method*), 34, 54, 140
- `get_list_attributes()` (*ListDataStruct_Interface method*), 28, 49
- `get_logopti()` (*_OptiProjectLoader method*), 134
- `get_logopti()` (*OptiHistoric method*), 78, 86
- `get_M2P()` (*OptimizerSettings method*), 80, 85
- `get_M2P()` (*SensitivityParameters method*), 17, 22
- `get_max_value()` (*Integer_OptimizationVariable method*), 79, 84
- `get_max_value()` (*Real_OptimizationVariable method*), 79, 84
- `get_meta()` (*Data method*), 32, 52, 138
- `get_min_value()` (*Integer_OptimizationVariable method*), 79, 84
- `get_min_value()` (*Real_OptimizationVariable method*), 79, 84
- `get_nadir_point()` (*EvolutionaryConvergence method*), 67, 69
- `get_name()` (*Base_Option method*), 40, 58
- `get_name()` (*FastObjCons method*), 65, 66, 82
- `get_name()` (*InterfaceObjCons method*), 66, 82
- `get_name()` (*Onclick_animate method*), 100, 107, 152
- `get_name()` (*Onclick_changeSymbol method*), 101, 108, 153
- `get_name()` (*Onclick_copySomething method*), 101, 108, 153
- `get_name()` (*Onclick_delete method*), 102, 108, 153
- `get_name()` (*Onclick_exportCollection method*), 102, 109, 154
- `get_name()` (*Onclick_exportToTxt method*), 103, 109, 154
- `get_name()` (*Onclick_exportTrace method*), 103, 109, 154
- `get_name()` (*Onclick_extractPareto method*), 104, 110, 155
- `get_name()` (*Onclick_measure method*), 105, 110, 142, 155
- `get_name()` (*Onclick_removeTrace method*), 105, 110, 155
- `get_name()` (*Onclick_representDevice method*), 106, 111, 152
- `get_name()` (*Onclick_tojson method*), 106, 111, 156
- `get_name()` (*Onselect_highlight method*), 114, 116, 158
- `get_name()` (*Onselect_newTrace method*), 115, 116, 159
- `get_name()` (*Onselect_splitTrace method*), 115, 117, 159
- `get_name_selected()` (*Widget_listWithSearch method*), 123, 126, 149
- `get_nb_objectives()` (*EvolutionaryConvergence method*), 67, 69
- `get_nbr_elements()` (*ListDataStruct method*), 20, 29, 50
- `get_nbr_elements()` (*Performance_ListDataStruct method*), 30, 51
- `get_nbr_objectives()` (*_OptiProjectLoader method*), 134
- `get_ND_pareto()` (*in module optimeed.core*), 46
- `get_ND_pareto()` (*in module optimeed.core.tools*), 44
- `get_new_index()` (*Onclick_representDevice.DataInformationVisuals method*), 105, 111, 152
- `get_number_of_contours()` (*ContourPlot method*), 36, 56, 143
- `get_number_of_elements()` (*_AnimationTrace method*), 97
- `get_number_of_points()` (*Data method*), 32, 52, 139
- `get_object_attrs()` (*in module optimeed.core*), 46
- `get_object_attrs()` (*in module optimeed.core.tools*), 44
- `get_objectives()` (*OptimizerSettings method*), 80, 85
- `get_opengl_options()` (*DeviceDrawerInterface method*), 118, 121, 128, 151
- `get_optialgorithm()` (*OptimizerSettings method*), 80, 85
- `get_option_name()` (*Option_class method*), 19, 41, 59
- `get_option_value()` (*Option_class method*), 19, 41, 59
- `get_optivariabes()` (*OptimizerSettings method*), 80, 85
- `get_optivariabes()` (*SensitivityParameters method*), 18, 22
- `get_paramvalues()` (*SensitivityParameters method*), 18, 22
- `get_pareto_at_step()` (*EvolutionaryConvergence method*), 67, 69
- `get_pareto_convergence()` (*EvolutionaryConvergence method*), 67, 69
- `get_path_to_inkscape()` (*in module optimeed.core*), 62
- `get_path_to_inkscape()` (*in module optimeed.core.inkscape_manager*), 37
- `get_permutations()` (*Data method*), 33, 53, 139
- `get_PhysToMaths()` (*Binary_OptimizationVariable method*), 79, 84

`get_PhysToMaths()` (*Integer_OptimizationVariable method*), 79, 84
`get_PhysToMaths()` (*OptimizationVariable method*), 78
`get_PhysToMaths()` (*Real_OptimizationVariable method*), 79, 84
`get_plot_data()` (*Data method*), 33, 52, 139
`get_plot_data()` (*GridPlot_Generic method*), 36, 56
`get_plot_data()` (*ScatterPlot3 method*), 36, 56, 143
`get_plot_meta()` (*Data method*), 33, 52, 139
`get_point()` (*TraceVisual method*), 93, 146
`get_recursive_attrs()` (*in module optimeed.core*), 46
`get_recursive_attrs()` (*in module optimeed.core.tools*), 44
`get_reference_device()` (*Parametric_parameter method*), 16, 21
`get_rows_indices()` (*OptiHistoric_LogParams method*), 78, 85
`get_scalar_convergence_evolution()` (*EvolutionaryConvergence method*), 67, 69
`get_sensitivity_problem()` (*in module optimeed consolidate*), 22
`get_sensitivity_problem()` (*in module optimeed consolidate.sensitivity_analysis*), 18
`get_shown_entries()` (*Widget_tableWithSearch method*), 125, 127, 151
`get_styles()` (*in module optimeed.core.ansi2html.style*), 26
`get_symbol()` (*Data method*), 33, 53, 140
`get_symbol()` (*TraceVisual method*), 92, 144
`get_symbolOutline()` (*Data method*), 32, 52, 138
`get_symbolPens()` (*TraceVisual method*), 93, 145
`get_symbolsSize()` (*Data method*), 32, 52, 138
`get_text_to_write()` (*ContextHandler method*), 118
`get_total_nbr_elements()` (*Performance_ListDataStruct method*), 30, 50
`get_trace()` (*Graph method*), 34, 54
`get_trace()` (*GraphVisual method*), 88, 147
`get_trace()` (*Widget_graphsVisualLite method*), 94, 96, 148
`get_type_class()` (*in module optimeed.core*), 61
`get_type_class()` (*in module optimeed.core.myjson*), 39
`get_value()` (*Base_Option method*), 40, 58
`get_values()` (*Parametric_minmax method*), 16, 21
`get_wgGraphs()` (*_PlotHolders method*), 132, 142
`get_wgGraphs()` (*in module optimeed.visualize*), 143
`get_wgGraphs()` (*in module optimeed.visualize.fastPlot*), 133
`get_wgGraphs()` (*WindowHolders method*), 132, 142
`get_widget()` (*Represent_brut_attributes method*), 107, 112, 156
`get_widget()` (*Represent_image method*), 107, 111, 156
`get_widget()` (*Represent_lines method*), 106, 112, 156
`get_widget()` (*Represent_opengl method*), 107, 111, 156
`get_widget()` (*Represent_txt_function method*), 106, 112, 156
`get_width()` (*Data method*), 32, 52, 139
`get_x()` (*Data method*), 32, 52, 138
`get_x_label()` (*Data method*), 33, 53, 140
`get_xlim()` (*Data method*), 32, 52, 138
`get_y()` (*Data method*), 32, 52, 138
`get_y_label()` (*Data method*), 33, 53, 140
`get_ylim()` (*Data method*), 32, 52, 138
`getAmb3()` (*MaterialRenderingProperties method*), 119, 121, 128, 151
`getCurrentShow()` (*in module optimeed.core*), 47
`getCurrentShow()` (*in module optimeed.core.commonImport*), 31
`getDif3()` (*MaterialRenderingProperties method*), 118, 121, 128, 151
`GetEar()` (*in module optimeed.visualize.widgets.openGL.triangulate_polygon*), 120
`getExecPath()` (*in module optimeed.core*), 61
`getExecPath()` (*in module optimeed.core.myjson*), 39
`getLength()` (*MultiList method*), 68
`getLineInfo()` (*in module optimeed.core*), 46
`getLineInfo()` (*in module optimeed.core.tools*), 44
`getPath_workspace()` (*in module optimeed consolidate*), 21
`getPath_workspace()` (*in module optimeed.core*), 46, 47
`getPath_workspace()` (*in module optimeed.core.tools*), 44
`getShin()` (*MaterialRenderingProperties method*), 119, 121, 128, 151
`getSpec3()` (*MaterialRenderingProperties method*), 118, 121, 128, 151
`getTxt()` (*Represent_txt_function method*), 106, 112, 156
`Graph` (*class in optimeed.core*), 54
`Graph` (*class in optimeed.core.graphs*), 34
`graph_clicked()` (*OnClick_animate method*), 100, 107, 152
`graph_clicked()` (*OnClick_changeSymbol method*), 101, 108, 152
`graph_clicked()` (*OnClick_copySomething method*), 101, 108, 153
`graph_clicked()` (*OnClick_delete method*), 102, 108, 153
`graph_clicked()` (*OnClick_exportCollection*

- method), 102, 109, 153
- graph_clicked() (*OnClick_exportToTxt method*), 103, 109, 154
- graph_clicked() (*OnClick_exportTrace method*), 103, 109, 154
- graph_clicked() (*OnClick_extractPareto method*), 104, 110, 154
- graph_clicked() (*OnClick_measure method*), 104, 110, 141, 155
- graph_clicked() (*OnClick_removeTrace method*), 105, 110, 155
- graph_clicked() (*OnClick_representDevice method*), 105, 111, 152
- graph_clicked() (*OnClick_tjson method*), 106, 111, 155
- Graphs (class in *optimeed.core*), 54
- Graphs (class in *optimeed.core.graphs*), 34
- Graphs (class in *optimeed.visualize*), 140
- GraphVisual (class in *optimeed.visualize*), 146
- GraphVisual (class in *optimeed.visualize.graphs.graphVisual*), 86
- GREEN (text_format attribute), 43, 45
- Green_material (in module *optimeed.visualize*), 152
- Green_material (in module *optimeed.visualize.widgets*), 129
- Green_material (in module *optimeed.visualize.widgets.openGL*), 121
- Green_material (in module *optimeed.visualize.widgets.openGL.materials*), 119
- grid_off() (*GraphVisual method*), 88, 147
- griddata_found (in module *optimeed.core*), 55
- griddata_found (in module *optimeed.core.graphs3*), 36
- GridPlot_Generic (class in *optimeed.core*), 56
- GridPlot_Generic (class in *optimeed.core.graphs3*), 36
- group() (*Plugin_listWithSearch method*), 123
- group() (*Plugin_tableWithSearch method*), 125
- ## H
- has_matplotlib (in module *optimeed.visualize.graphs.colormap_pyqtgraph*), 86
- has_scipy (in module *optimeed.core*), 59
- has_scipy (in module *optimeed.core.additional_tools*), 26
- hasPlotly (in module *optimeed.visualize*), 143
- hasPlotly (in module *optimeed.visualize.fastPlot3*), 133
- hide() (*TraceVisual method*), 92, 145
- hide_axes() (*GraphVisual method*), 87, 147
- hide_points() (*TraceVisual method*), 91, 144
- hideRow() (*Widget_tableWithSearch method*), 125, 127, 150
- HowToPlotGraph (class in *optimeed.core*), 56
- HowToPlotGraph (class in *optimeed.core.linkDataGraph*), 37
- hvRecursive() (*HyperVolume method*), 67
- HyperVolume (class in *optimeed.optimize.optiAlgorithms.convergence.hypervolume*), 67
- hypervolume_per_step (*EvolutionaryConvergence attribute*), 67, 69
- ## I
- icon() (*Plugin_listWithSearch method*), 123
- icon() (*Plugin_tableWithSearch method*), 125
- includeFile() (*Plugin_listWithSearch method*), 123
- includeFile() (*Plugin_tableWithSearch method*), 125
- indentParagraph() (in module *optimeed.core*), 46, 48
- indentParagraph() (in module *optimeed.core.tools*), 44
- index (*SensitivityResults attribute*), 17
- index() (in module *optimeed.core.ansi2html.style*), 25
- index2() (in module *optimeed.core.ansi2html.style*), 25
- initialize() (*ContextHandler method*), 118
- initialize() (*ConvergenceTerminationCondition method*), 74
- initialize() (*MaxTimeTerminationCondition method*), 73
- initialize() (*Monobjective_PSO method*), 72, 77, 83
- initialize() (*MultiObjective_GA method*), 75, 76, 82
- initialize() (*NLOpt_Algorithm method*), 71
- initialize() (*Plugin_listWithSearch method*), 123
- initialize() (*Plugin_tableWithSearch method*), 125
- initialize() (*SeveralTerminationCondition method*), 74
- initialize_output_collection() (*Parametric_analysis method*), 17, 21
- initializeGL() (*Widget_openGL method*), 124, 127, 150
- inkscape_svg_to_pdf() (in module *optimeed.core*), 62
- inkscape_svg_to_pdf() (in module *optimeed.core.inkscape_manager*), 37
- inkscape_svg_to_pdf() (in module *optimeed.visualize*), 143
- inkscape_svg_to_png() (in module *optimeed.core*), 62
- inkscape_svg_to_png() (in module *optimeed.core.inkscape_manager*), 37

`inkscape_svg_to_png()` (in module `optimeed.visualize`), 143
`inkscape_version` (in module `optimeed.core`), 62
`inkscape_version` (in module `optimeed.core.inkscape_manager`), 37
`Integer_OptimizationVariable` (class in `optimeed.optimize`), 84
`Integer_OptimizationVariable` (class in `optimeed.optimize.optiVariable`), 79
`integrate()` (in module `optimeed.core`), 60
`integrate()` (in module `optimeed.core.additional_tools`), 27
`intensify()` (in module `optimeed.core.ansi2html.style`), 25
`InterfaceCharacterization` (class in `optimeed.optimize`), 81
`InterfaceCharacterization` (class in `optimeed.optimize.characterization`), 63
`InterfaceCharacterization` (class in `optimeed.optimize.characterization.interfaceCharacterization`), 63
`InterfaceConvergence` (class in `optimeed.optimize.optiAlgorithms.convergence`), 69
`InterfaceConvergence` (class in `optimeed.optimize.optiAlgorithms.convergence.interfaceConvergence`), 68
`InterfaceMathsToPhysics` (class in `optimeed.optimize`), 82
`InterfaceMathsToPhysics` (class in `optimeed.optimize.mathsToPhysics`), 65
`InterfaceMathsToPhysics` (class in `optimeed.optimize.mathsToPhysics.interfaceMathsToPhysics`), 64
`InterfaceObjCons` (class in `optimeed.optimize`), 82
`InterfaceObjCons` (class in `optimeed.optimize.objAndCons`), 66
`InterfaceObjCons` (class in `optimeed.optimize.objAndCons.interfaceObjCons`), 66
`interpolate()` (`fast_LUT_interpolation` method), 27, 59
`interpolate_table()` (in module `optimeed.core`), 59
`interpolate_table()` (in module `optimeed.core.additional_tools`), 27
`InTriangle()` (in module `optimeed.visualize.widgets.openGL.triangulate_polygon`), 120
`is_empty()` (`AnimationGUI` method), 98
`is_empty()` (`Graphs` method), 36, 55, 141
`IsClockwise()` (in module `optimeed.visualize.widgets.openGL.triangulate_polygon`), 120
`isContainer()` (`Plugin_listWithSearch` method), 123
`isContainer()` (`Plugin_tableWithSearch` method), 125
`IsConvex()` (in module `optimeed.visualize.widgets.openGL.triangulate_polygon`), 120
`isInitialized()` (`Plugin_listWithSearch` method), 123
`isInitialized()` (`Plugin_tableWithSearch` method), 125
`isNonePrintMessage()` (in module `optimeed.core`), 46
`isNonePrintMessage()` (in module `optimeed.core.tools`), 44
`isOnWindows` (in module `optimeed.visualize.graphs.pyqtgraphRedefine`), 88
`isScattered()` (`Data` method), 34, 53, 140
`json_to_obj()` (in module `optimeed.core`), 48, 61
`json_to_obj()` (in module `optimeed.core.myjson`), 39
`json_to_obj_safe()` (in module `optimeed.core`), 48, 61
`json_to_obj_safe()` (in module `optimeed.core.myjson`), 40

K

`keyboard_push_action()` (`DeviceDrawerInterface` method), 118, 121, 128, 151
`keyboardPushAction()` (`ContextHandler` method), 118
`keyboardReleaseAction()` (`ContextHandler` method), 118
`keyPressEvent()` (`MainWindow` method), 133, 138
`keyPressEvent()` (`Widget_graphsVisualLite` method), 94, 96, 149
`keyPressEvent()` (`Widget_openGL` method), 124, 127, 150
`KWARGS_ALGO` (`MultiObjective_GA` attribute), 75, 76, 82

L

`last_step()` (`EvolutionaryConvergence` method), 67, 69
`launch_embarrassingly_parallel_sensitivity()` (in module `optimeed consolidate`), 23
`launch_embarrassingly_parallel_sensitivity()` (in module `optimeed consolidate.sensitivity_analysis`), 18
`launch_optimization()` (`OptimizationDisplay` method), 130, 137
`leastSquare()` (in module `optimeed consolidate`), 21

- leastSquare() (in module *optimeed consolidate.fit*), 15
- level() (in module *optimeed.core.ansi2html.style*), 25
- link_axes() (*Widget_graphsVisualLite* method), 94, 96, 148
- LinkDataGraph (class in *optimeed.core*), 56
- LinkDataGraph (class in *optimeed.core.linkDataGraph*), 37
- linkify() (in module *optimeed.core.ansi2html.converter*), 24
- linkXToGraph() (*GraphVisual* method), 88, 147
- linspace() (in module *optimeed.core*), 59
- linspace() (in module *optimeed.core.additional_tools*), 27
- list_of_optimization_variables (*Sensitivity-Parameters* attribute), 17, 22
- ListDataStruct (class in *optimeed consolidate*), 19
- ListDataStruct (class in *optimeed.core*), 49
- ListDataStruct (class in *optimeed.core.collection*), 29
- ListDataStruct_Interface (class in *optimeed.core*), 49
- ListDataStruct_Interface (class in *optimeed.core.collection*), 28
- log_after_evaluation() (*OptiHistoric* method), 78, 85
- log_mode() (*GraphVisual* method), 88, 147
- ## M
- main() (in module *optimeed.core.ansi2html.converter*), 25
- MainWindow (class in *optimeed.visualize*), 138
- MainWindow (class in *optimeed.visualize.mainWindow*), 133
- map_index() (*_AnimationTrace* method), 97
- map_vt100_box_code() (in module *optimeed.core.ansi2html.converter*), 24
- MaterialRenderingProperties (class in *optimeed.visualize*), 151
- MaterialRenderingProperties (class in *optimeed.visualize.widgets*), 128
- MaterialRenderingProperties (class in *optimeed.visualize.widgets.openGL*), 121
- MaterialRenderingProperties (class in *optimeed.visualize.widgets.openGL.materials*), 118
- MathsToPhysics (class in *optimeed.optimize*), 81
- MathsToPhysics (class in *optimeed.optimize.mathsToPhysics*), 65
- MathsToPhysics (class in *optimeed.optimize.mathsToPhysics.mathsToPhysics*), 64
- matplotlib_colormap_to_pg_colormap() (in module *optimeed.visualize.graphs.colormap_pyqtgraph*), 86
- MaxTimeTerminationCondition (class in *optimeed.optimize.optiAlgorithms.multiObjective_GA*), 73
- merge() (*Graphs* method), 35, 55, 141
- merge() (*ListDataStruct* method), 20, 29, 50
- merge_two_dicts() (in module *optimeed.core*), 47
- merge_two_dicts() (in module *optimeed.core.tools*), 45
- MeshPlot (class in *optimeed.core*), 56
- MeshPlot (class in *optimeed.core.graphs3*), 36
- MeshPlot (class in *optimeed.visualize*), 143
- meshPolygon() (in module *optimeed.visualize.widgets.openGL.triangulate_polygon*), 120
- minimumSizeHint() (*Widget_openGL* method), 124, 127, 150
- MODE_LIGHT (in module *optimeed.visualize.widgets.openGL.contextHandler*), 117
- MODE_ROTATION (in module *optimeed.visualize.widgets.openGL.contextHandler*), 117
- MODE_ZOOM (in module *optimeed.visualize.widgets.openGL.contextHandler*), 117
- modify_paintElems() (*TraceVisual._ModifiedPaintElem* method), 91, 144
- MODULE_TAG (in module *optimeed.core*), 60
- MODULE_TAG (in module *optimeed.core.myjson*), 39
- Monobjective_PSO (class in *optimeed.optimize*), 83
- Monobjective_PSO (class in *optimeed.optimize.optiAlgorithms*), 76
- Monobjective_PSO (class in *optimeed.optimize.optiAlgorithms.monobjective_PSO*), 72
- mouseClicAction() (*ContextHandler* method), 118
- mouseMotionAction() (*ContextHandler* method), 118
- mouseMoveEvent() (*Widget_openGL* method), 124, 127, 150
- mousePressEvent() (*Widget_openGL* method), 124, 127, 150
- mouseReleaseEvent() (*Widget_menuButton* method), 124, 127, 148, 150
- mouseWheelAction() (*ContextHandler* method), 117
- MultiList (class in *optimeed.optimize.optiAlgorithms.convergence.hypervolume*), 68
- MultiList.Node (class in *optimeed.optimize.optiAlgorithms.convergence.hypervolume*), 68
- MultiObjective_GA (class in *optimeed.optimize*), 82

MultiObjective_GA (class in *optimeed.optimize.optiAlgorithms*), 76

MultiObjective_GA (class in *optimeed.optimize.optiAlgorithms.multiObjective_GA*), 75

my_callback() (*MyMultiprocessEvaluator* method), 75

my_fft() (in module *optimeed.core*), 59

my_fft() (in module *optimeed.core.additional_tools*), 27

my_fourier() (in module *optimeed.core*), 60

my_fourier() (in module *optimeed.core.additional_tools*), 27

myAxis (class in *optimeed.visualize.graphs.pyqtgraphRedefine*), 90

MyGenerator (class in *optimeed.optimize.optiAlgorithms.multiObjective_GA*), 73

myGraphicsLayout (class in *optimeed.visualize.graphs.pyqtgraphRedefine*), 89

myGraphicsLayoutWidget (class in *optimeed.visualize*), 143

myGraphicsLayoutWidget (class in *optimeed.visualize.graphs.pyqtgraphRedefine*), 89

myItemSample (class in *optimeed.visualize.graphs.pyqtgraphRedefine*), 89

myLabelItem (class in *optimeed.visualize.graphs.pyqtgraphRedefine*), 90

myLegend (class in *optimeed.visualize.graphs.pyqtgraphRedefine*), 90

MyMapEvaluator (class in *optimeed.optimize.optiAlgorithms.multiObjective_GA*), 74

MyMapEvaluator (class in *optimeed.optimize.optiAlgorithms.pyswarm*), 70

MyMapEvaluator (class in *optimeed.optimize.optiAlgorithms.pyswarm.pso*), 69

MyMultiprocessEvaluator (class in *optimeed.optimize.optiAlgorithms.multiObjective_GA*), 74

MyMultiprocessEvaluator (class in *optimeed.optimize.optiAlgorithms.pyswarm*), 70

MyMultiprocessEvaluator (class in *optimeed.optimize.optiAlgorithms.pyswarm.pso*), 69

MyProblem (class in *optimeed.optimize.optiAlgorithms.multiObjective_GA*), 73

MyText (class in *optimeed.visualize.widgets.openGL.contextHandler*), 117

myWindows (in module *optimeed.visualize*), 142

myWindows (in module *optimeed.visualize.fastPlot*), 132

N

name (*Option_bool* attribute), 41, 58

name (*Option_dict* attribute), 41, 59

name (*Option_float* attribute), 41, 59

name (*Option_int* attribute), 19, 41, 58

name (*Option_str* attribute), 41, 58

name() (*Plugin_listWithSearch* method), 123

name() (*Plugin_tableWithSearch* method), 125

new_figure() (*WindowHolders* method), 132, 142

new_plot() (*_PlotHolders* method), 132, 142

new_plot() (*WindowHolders* method), 132, 142

next_frame() (*AnimationGUI* method), 97

NLOpt_Algorithm (class in *optimeed.optimize.optiAlgorithms.NLOpt_Algorithm*), 71

normalize() (in module *optimeed.visualize.widgets.openGL.quaternions*), 120

NUMBER_OF_CORES (*Monobjective_PSO* attribute), 72, 76, 83

NUMBER_OF_CORES (*MultiObjective_GA* attribute), 75, 76, 82

NUMBER_OF_CORES (*Parametric_analysis* attribute), 17, 21

NUMBER_OF_MODES (in module *optimeed.visualize.widgets.openGL.contextHandler*), 117

O

obj_to_json() (in module *optimeed.core*), 48, 61

obj_to_json() (in module *optimeed.core.myjson*), 40

objectives (*OptiHistoric._pointData* attribute), 77, 85

objectives_per_step (*EvolutionaryConvergence* attribute), 67, 69

on_click() (*Widget_graphsVisualLite* method), 94, 95, 148

on_update_signal() (*Widget_lineDrawer* method), 122, 126, 150

Onclick_animate (class in *optimeed.visualize*), 152

Onclick_animate (class in *optimeed.visualize.onclick*), 107

Onclick_animate (class in *optimeed.visualize.onclick.onclick_animate*), 100

Onclick_changeSymbol (class in optimeed.visualize), 152	Onclick_removeTrace (class in optimeed.visualize), 155
Onclick_changeSymbol (class in optimeed.visualize.onclick), 107	Onclick_removeTrace (class in optimeed.visualize.onclick), 110
Onclick_changeSymbol (class in optimeed.visualize.onclick.onclick_changeSymbol), 101	Onclick_removeTrace (class in optimeed.visualize.onclick.onclick_removeTrace), 105
Onclick_copySomething (class in optimeed.visualize), 153	Onclick_representDevice (class in optimeed.visualize), 152
Onclick_copySomething (class in optimeed.visualize.onclick), 108	Onclick_representDevice (class in optimeed.visualize.onclick), 110
Onclick_copySomething (class in optimeed.visualize.onclick.onclick_copySomething), 101	Onclick_representDevice (class in optimeed.visualize.onclick.onclick_representDevice), 105
Onclick_delete (class in optimeed.visualize), 153	Onclick_representDevice.DataInformationVisuals (class in optimeed.visualize), 152
Onclick_delete (class in optimeed.visualize.onclick), 108	Onclick_representDevice.DataInformationVisuals (class in optimeed.visualize.onclick), 111
Onclick_delete (class in optimeed.visualize.onclick.onclick_delete), 102	Onclick_representDevice.DataInformationVisuals (class in optimeed.visualize.onclick.onclick_representDevice), 105
Onclick_exportCollection (class in optimeed.visualize), 153	Onclick_tojson (class in optimeed.visualize), 155
Onclick_exportCollection (class in optimeed.visualize.onclick), 108	Onclick_tojson (class in optimeed.visualize.onclick), 111
Onclick_exportCollection (class in optimeed.visualize.onclick.onclick_exportCollection), 102	Onclick_tojson (class in optimeed.visualize.onclick.onclick_tojson), 106
Onclick_exportToTxt (class in optimeed.visualize), 154	OnClickInterface (class in optimeed.visualize), 156
Onclick_exportToTxt (class in optimeed.visualize.onclick), 109	OnClickInterface (class in optimeed.visualize.onclick), 111
Onclick_exportToTxt (class in optimeed.visualize.onclick.onclick_exportToTxt), 103	OnClickInterface (class in optimeed.visualize.onclick.onclickInterface), 100
Onclick_exportTrace (class in optimeed.visualize), 154	Onselect_highlight (class in optimeed.visualize), 158
Onclick_exportTrace (class in optimeed.visualize.onclick), 109	Onselect_highlight (class in optimeed.visualize.selector), 116
Onclick_exportTrace (class in optimeed.visualize.onclick.onclick_exportTrace), 103	Onselect_highlight (class in optimeed.visualize.selector.onselect_highlight), 114
Onclick_extractPareto (class in optimeed.visualize), 154	Onselect_newTrace (class in optimeed.visualize), 158
Onclick_extractPareto (class in optimeed.visualize.onclick), 109	Onselect_newTrace (class in optimeed.visualize.selector), 116
Onclick_extractPareto (class in optimeed.visualize.onclick.onclick_extractPareto), 104	Onselect_newTrace (class in optimeed.visualize.selector.onselect_newTrace), 115
Onclick_measure (class in optimeed.visualize), 141, 155	Onselect_splitTrace (class in optimeed.visualize), 159
Onclick_measure (class in optimeed.visualize.onclick), 110	Onselect_splitTrace (class in optimeed.visualize.selector), 116
Onclick_measure (class in optimeed.visualize.onclick.onclick_measure), 104	Onselect_splitTrace (class in optimeed.visualize.selector.onselect_splitTrace), 115

OnselectInterface (class in *optimeed.visualize*), 158

OnselectInterface (class in *optimeed.visualize.selector*), 116

OnselectInterface (class in *optimeed.visualize.selector.onselectInterface*), 114

OPTI_ALGORITHM (*MultiObjective_GA* attribute), 75, 76, 82

OptiHistoric (class in *optimeed.optimize*), 85

OptiHistoric (class in *optimeed.optimize.optiHistoric*), 77

OptiHistoric._LogParams (class in *optimeed.optimize*), 85

OptiHistoric._LogParams (class in *optimeed.optimize.optiHistoric*), 77

OptiHistoric._pointData (class in *optimeed.optimize*), 85

OptiHistoric._pointData (class in *optimeed.optimize.optiHistoric*), 77

optimeed (module), 15

optimeed consolidate (module), 15

optimeed consolidate.fit (module), 15

optimeed consolidate.parametric_analysis (module), 16

optimeed consolidate.sensitivity_analysis (module), 17

optimeed consolidate.sensitivity_analysis_optimized (module), 19

optimeed core (module), 23

optimeed core.additional_tools (module), 26

optimeed core.ansi2html (module), 23

optimeed core.ansi2html.converter (module), 23

optimeed core.ansi2html.style (module), 25

optimeed core.ansi2html.util (module), 26

optimeed core.collection (module), 28

optimeed core.color_palette (module), 31

optimeed core.commonImport (module), 31

optimeed core.graphs (module), 32

optimeed core.graphs3 (module), 36

optimeed core.inkscape_manager (module), 37

optimeed core.linkDataGraph (module), 37

optimeed core.myjson (module), 39

optimeed core.options (module), 40

optimeed core.tikzTranslator (module), 42

optimeed core.tools (module), 43

optimeed optimize (module), 62

optimeed optimize.characterization (module), 63

optimeed optimize.characterization.characterization (module), 63

optimeed optimize.characterization.interfaceCharacterization (module), 63

optimeed optimize.mathsToPhysics (module), 64

optimeed optimize.mathsToPhysics.interfaceMathsToPhysics (module), 64

optimeed optimize.mathsToPhysics.mathsToPhysics (module), 64

optimeed optimize.objAndCons (module), 65

optimeed optimize.objAndCons.fastObjCons (module), 65

optimeed optimize.objAndCons.interfaceObjCons (module), 66

optimeed optimize.optiAlgorithms (module), 66

optimeed optimize.optiAlgorithms.algorithmInterface (module), 72

optimeed optimize.optiAlgorithms.convergence (module), 66

optimeed optimize.optiAlgorithms.convergence.evolution (module), 66

optimeed optimize.optiAlgorithms.convergence.hyperparameter (module), 67

optimeed optimize.optiAlgorithms.convergence.intermediate (module), 68

optimeed optimize.optiAlgorithms.monobjective_PSO (module), 72

optimeed optimize.optiAlgorithms.multiObjective_GA (module), 73

optimeed optimize.optiAlgorithms.NLOpt_Algorithm (module), 71

optimeed optimize.optiAlgorithms.pyswarm (module), 69

optimeed optimize.optiAlgorithms.pyswarm.pso (module), 69

optimeed optimize.optiHistoric (module), 77

optimeed optimize.optimizer (module), 80

optimeed optimize.optiVariable (module), 78

optimeed visualize (module), 86

optimeed visualize.displayCollections (module), 129

optimeed visualize.displayOptimization (module), 130

optimeed visualize.displaySensitivity (module), 130

optimeed visualize.fastPlot (module), 132

optimeed visualize.fastPlot3 (module), 133

optimeed visualize.graphs (module), 86

optimeed visualize.graphs.colormap_pyqtgraph (module), 86

optimeed visualize.graphs.graphVisual (module), 86

optimeed.visualize.graphs.pyqtgraphRedefine (module), 88
 optimeed.visualize.graphs.traceVisual (module), 91
 optimeed.visualize.graphs.widget_graphsVisual (module), 93
 optimeed.visualize.mainWindow (module), 133
 optimeed.visualize.onclick (module), 96
 optimeed.visualize.onclick.animation_example (module), 98
 optimeed.visualize.onclick.animationGUI (module), 96
 optimeed.visualize.onclick.collectionExportGUI (module), 99
 optimeed.visualize.onclick.onclick_animation (module), 100
 optimeed.visualize.onclick.onclick_changeStyle (module), 101
 optimeed.visualize.onclick.onclick_copyScreenshot (module), 101
 optimeed.visualize.onclick.onclick_delete (module), 102
 optimeed.visualize.onclick.onclick_exportCollection (module), 102
 optimeed.visualize.onclick.onclick_exportFile (module), 103
 optimeed.visualize.onclick.onclick_exportFrame (module), 103
 optimeed.visualize.onclick.onclick_extraOptions (module), 104
 optimeed.visualize.onclick.onclick_measure (module), 104
 optimeed.visualize.onclick.onclick_removeFrame (module), 105
 optimeed.visualize.onclick.onclick_representDevice (module), 105
 optimeed.visualize.onclick.onclick_tojson (module), 106
 optimeed.visualize.onclick.onclickInterface (module), 100
 optimeed.visualize.onclick.representDeviceExample (module), 106
 optimeed.visualize.process_mainloop (module), 133
 optimeed.visualize.selector (module), 114
 optimeed.visualize.selector.onselect_highlight (module), 114
 optimeed.visualize.selector.onselect_newOption (module), 114
 optimeed.visualize.selector.onselect_splitFrame (module), 115
 optimeed.visualize.selector.onselectInterOption (module), 114
 optimeed.visualize.viewOptimizationResults (module), 134
 optimeed.visualize.widgets (module), 117
 optimeed.visualize.widgets.openGL (module), 117
 optimeed.visualize.widgets.openGL.contextHandler (module), 117
 optimeed.visualize.widgets.openGL.deviceDrawerInterface (module), 118
 optimeed.visualize.widgets.openGL.materials (module), 118
 optimeed.visualize.widgets.openGL.openGL_library (module), 119
 optimeed.visualize.widgets.openGL.quaternions (module), 120
 optimeed.visualize.widgets.openGL.triangulate_polygon (module), 120
 optimeed.visualize.widgets.widget_doubleSlider (module), 122
 optimeed.visualize.widgets.widget_image (module), 122
 optimeed.visualize.widgets.widget_lineDrawer (module), 122
 optimeed.visualize.widgets.widget_listWithSearch (module), 123
 optimeed.visualize.widgets.widget_listWithSearchplus (module), 123
 optimeed.visualize.widgets.widget_menuButton (module), 124
 optimeed.visualize.widgets.widget_openGL (module), 124
 optimeed.visualize.widgets.widget_tableWithSearch (module), 124
 optimeed.visualize.widgets.widget_tableWithSearchplus (module), 125
 optimeed.visualize.widgets.widget_text (module), 126
 OptimizationDisplay (class in optimeed.visualize), 136
 OptimizationDisplay (class in optimeed.visualize.displayOptimization), 130
 OptimizationVariable (class in optimeed.optimize.optiVariable), 78
 OptimizerSettings (class in optimeed.optimize), 85
 OptimizerSettings (class in optimeed.optimize.optimizer), 80
 Option_bool (class in optimeed.core), 58
 Option_bool (class in optimeed.core.options), 40
 Option_class (class in optimeed consolidate), 19
 Option_class (class in optimeed.core), 59
 Option_frameclass (class in optimeed.core.options), 41
 Option_dict (class in optimeed.core), 59
 Option_dict (class in optimeed.core.options), 41
 Option_float (class in optimeed.core), 58

Option_float (class in *optimeed.core.options*), 41
 Option_int (class in *optimeed.consolidate*), 19
 Option_int (class in *optimeed.core*), 58
 Option_int (class in *optimeed.core.options*), 41
 Option_str (class in *optimeed.core*), 58
 Option_str (class in *optimeed.core.options*), 41
 options_bool (*Option_class* attribute), 19, 41, 59
 options_dict (*Option_class* attribute), 19, 41, 59
 options_float (*Option_class* attribute), 19, 41, 59
 options_int (*Option_class* attribute), 19, 41, 59
 options_str (*Option_class* attribute), 19, 41, 59
 order_lists() (in module *optimeed.core*), 47
 order_lists() (in module *optimeed.core.tools*), 45

P

paint() (*_LineItem* method), 104
 paint() (*myItemSample* method), 90
 paint() (*myLegend* method), 90
 paintEvent() (*Widget_lineDrawer* method), 123, 126, 150
 paintGL() (*Widget_openGL* method), 124, 127, 150
 param_values (*SensitivityParameters* attribute), 17, 22
 Parametric_analysis (class in *optimeed.consolidate*), 21
 Parametric_analysis (class in *optimeed.consolidate.parametric_analysis*), 17
 Parametric_Collection (class in *optimeed.consolidate*), 21
 Parametric_Collection (class in *optimeed.consolidate.parametric_analysis*), 16
 Parametric_minmax (class in *optimeed.consolidate*), 21
 Parametric_minmax (class in *optimeed.consolidate.parametric_analysis*), 16
 Parametric_parameter (class in *optimeed.consolidate*), 21
 Parametric_parameter (class in *optimeed.consolidate.parametric_analysis*), 16
 paramsToEvaluate (*SensitivityResults* attribute), 17
 paretos_per_step (*EvolutionaryConvergence* attribute), 67, 69
 partition() (in module *optimeed.core*), 60
 partition() (in module *optimeed.core.additional_tools*), 27
 pause_play() (*AnimationGUI* method), 97
 Performance_ListDataStruct (class in *optimeed.core*), 50
 Performance_ListDataStruct (class in *optimeed.core.collection*), 30

Pink_material (in module *optimeed.visualize*), 152
 Pink_material (in module *optimeed.visualize.widgets*), 129
 Pink_material (in module *optimeed.visualize.widgets.openGL*), 121
 Pink_material (in module *optimeed.visualize.widgets.openGL.materials*), 119
 plot() (in module *optimeed.visualize*), 142
 plot() (in module *optimeed.visualize.fastPlot*), 132
 Plot3D_Generic (class in *optimeed.core*), 55
 Plot3D_Generic (class in *optimeed.core.graphs3*), 36
 Plugin_listWithSearch (class in *optimeed.visualize.widgets.widget_listWithSearchplugin*), 123
 Plugin_tableWithSearch (class in *optimeed.visualize.widgets.widget_tableWithSearchplugin*), 125
 pol2cart() (in module *optimeed.core*), 59
 pol2cart() (in module *optimeed.core.additional_tools*), 27
 POPULATION_SIZE (*NLOpt_Algorithm* attribute), 71
 prepare() (*Ansi2HTMLConverter* method), 25, 26
 prepare_embarrassingly_parallel_sensitivity() (in module *optimeed.consolidate*), 23
 prepare_embarrassingly_parallel_sensitivity() (in module *optimeed.consolidate.sensitivity_analysis*), 18
 preProcess() (*HyperVolume* method), 68
 printIfShown() (in module *optimeed.core*), 46, 48, 51, 60, 62
 printIfShown() (in module *optimeed.core.tools*), 44
 printIfShown() (in module *optimeed.visualize*), 143
 process_qt_events() (in module *optimeed.visualize.process_mainloop*), 134
 produce_headers() (*Ansi2HTMLConverter* method), 25, 26
 pso() (in module *optimeed.optimize.optiAlgorithms.pyswarm*), 70
 pso() (in module *optimeed.optimize.optiAlgorithms.pyswarm.pso*), 70
 PURPLE (*text_format* attribute), 43, 45

Q

q_conjugate() (in module *optimeed.visualize.widgets.openGL.quaternions*), 120
 q_mult() (in module *optimeed.visualize.widgets.openGL.quaternions*), 120

- `q_to_axisangle()` (in module `optimeed.visualize.widgets.openGL.quaternions`), 120
`q_to_mat4()` (in module `optimeed.visualize.widgets.openGL.quaternions`), 120
`quicksort()` (in module `optimeed.core`), 60
`quicksort()` (in module `optimeed.core.additional_tools`), 27
`qv_mult()` (in module `optimeed.visualize.widgets.openGL.quaternions`), 120
- ## R
- `r_squared()` (in module `optimeed.consolidate.fit`), 16
`read_to_unicode()` (in module `optimeed.core.ansi2html.util`), 26
`Real_OptimizationVariable` (class in `optimeed.optimize`), 84
`Real_OptimizationVariable` (class in `optimeed.optimize.optiVariable`), 78
`reconstitute_signal()` (in module `optimeed.core`), 59
`reconstitute_signal()` (in module `optimeed.core.additional_tools`), 27
`RED` (text_format attribute), 43, 45
`Red_material` (in module `optimeed.visualize`), 152
`Red_material` (in module `optimeed.visualize.widgets`), 128
`Red_material` (in module `optimeed.visualize.widgets.openGL`), 121
`Red_material` (in module `optimeed.visualize.widgets.openGL.materials`), 119
`redraw()` (`ContextHandler` method), 118
`reevaluate_solutions()` (`_Evaluator` method), 81
`reformatXYtoList()` (in module `optimeed.visualize.widgets.openGL.triangulate_polygon`), 120
`refreshTraceList()` (`Widget_graphsVisual` method), 95, 96, 137, 149
`reinsert()` (`MultiList` method), 68
`remove()` (`MultiList` method), 68
`remove_collection()` (`CollectionDisplayer` method), 129, 135
`remove_collection()` (`LinkDataGraph` method), 37, 57
`remove_feature()` (`GraphVisual` method), 87, 147
`remove_forced_hide_row()` (`Widget_get_tableWithSearch` method), 125, 127, 150
`remove_graph()` (`Graphs` method), 35, 55, 141
`remove_module_tree_from_string()` (in module `optimeed.core`), 48, 62
`remove_module_tree_from_string()` (in module `optimeed.core.myjson`), 40
`remove_trace()` (`Graph` method), 34, 54
`remove_trace()` (`Graphs` method), 35, 55, 140
`reorder()` (`Performance_ListDataStruct` method), 30, 51
`Represent_brut_attributes` (class in `optimeed.visualize`), 156
`Represent_brut_attributes` (class in `optimeed.visualize.onclick`), 112
`Represent_brut_attributes` (class in `optimeed.visualize.onclick.representDevice_examples`), 107
`Represent_image` (class in `optimeed.visualize`), 156
`Represent_image` (class in `optimeed.visualize.onclick`), 111
`Represent_image` (class in `optimeed.visualize.onclick.representDevice_examples`), 107
`Represent_lines` (class in `optimeed.visualize`), 156
`Represent_lines` (class in `optimeed.visualize.onclick`), 112
`Represent_lines` (class in `optimeed.visualize.onclick.representDevice_examples`), 106
`Represent_opengl` (class in `optimeed.visualize`), 156
`Represent_opengl` (class in `optimeed.visualize.onclick`), 111
`Represent_opengl` (class in `optimeed.visualize.onclick.representDevice_examples`), 107
`Represent_txt_function` (class in `optimeed.visualize`), 156
`Represent_txt_function` (class in `optimeed.visualize.onclick`), 112
`Represent_txt_function` (class in `optimeed.visualize.onclick.representDevice_examples`), 106
`RepresentDeviceInterface` (class in `optimeed.visualize`), 152
`RepresentDeviceInterface` (class in `optimeed.visualize.onclick`), 107
`RepresentDeviceInterface` (class in `optimeed.visualize.onclick.onclick_representDevice`), 105
`reset()` (`_PlotHolders` method), 132, 142
`reset()` (`_State` method), 24
`reset()` (`AlgorithmInterface` method), 72
`reset()` (`CollectionExporterGUI` method), 100
`reset()` (`Graphs` method), 36, 55, 141
`reset()` (`MultiObjective_GA` method), 75, 76, 83
`reset()` (`TraceVisual._ModifiedPaintElem` method),

91, 144
 reset_all() (AnimationGUI method), 97
 reset_all_brushes() (TraceVisual method), 92, 145
 reset_all_symbolPens() (TraceVisual method), 93, 146
 reset_brush() (TraceVisual method), 92, 145
 reset_brushes() (TraceVisual method), 92, 145
 reset_data() (ListDataStruct method), 20, 29, 50
 reset_distance() (Onclick_measure method), 105, 110, 142, 155
 reset_graph() (Onclick_exportCollection method), 102, 109, 154
 reset_paintElem() (TraceVisual._ModifiedPaintElem method), 91, 144
 reset_symbol() (TraceVisual method), 92, 145
 reset_symbolPen() (TraceVisual method), 93, 146
 reset_symbolPens() (TraceVisual method), 93, 146
 resizeEvent() (myAxis method), 91
 resizeGL() (Widget_openGL method), 124, 127, 150
 resizeWindowAction() (ContextHandler method), 117
 rgetattr() (in module optimeed consolidate), 21
 rgetattr() (in module optimeed core), 46, 48, 60
 rgetattr() (in module optimeed core.tools), 44
 rsetattr() (in module optimeed consolidate), 21
 rsetattr() (in module optimeed core), 46, 60
 rsetattr() (in module optimeed core.tools), 44
 Rule (class in optimeed core.ansi2html.style), 25
 run() (AnimationGUI method), 98
 run() (MainWindow method), 133, 138
 run() (Parametric_analysis method), 17, 21
 run_optimization() (in module optimeed.optimize), 85
 run_optimization() (in module optimeed.optimize.optimizer), 81
 run_optimization_displayer() (in module optimeed.visualize.displayOptimization), 130

S

save() (AutosaveStruct method), 21, 28, 49
 save() (ListDataStruct method), 20, 29, 49
 save() (OptiHistoric method), 78, 85
 save() (Performance_ListDataStruct method), 30, 51
 SaveableObject (class in optimeed core), 61
 SaveableObject (class in optimeed core.myjson), 39
 ScatterPlot3 (class in optimeed core), 56
 ScatterPlot3 (class in optimeed core.graphs3), 36
 ScatterPlot3 (class in optimeed.visualize), 143
 SCHEME (in module optimeed core.ansi2html.style), 25
 select_folder_and_export() (Widget_graphsVisualLite method), 94, 95, 148
 selector_updated() (Onselect_highlight method), 114, 116, 158

selector_updated() (Onselect_newTrace method), 115, 116, 159
 selector_updated() (Onselect_splitTrace method), 115, 116, 159
 SensitivityDisplayer (class in optimeed.visualize), 135
 SensitivityDisplayer (class in optimeed.visualize.displaySensitivity), 131
 SensitivityParameters (class in optimeed consolidate), 22
 SensitivityParameters (class in optimeed consolidate.sensitivity_analysis), 17
 SensitivityResults (class in optimeed consolidate.sensitivity_analysis), 17
 sequence (in module optimeed.visualize.graphs.colormap_pyqtgraph), 86
 set_action_selector() (CollectionDisplayer method), 129, 135
 set_actionOnClick() (Widget_graphsVisualLite method), 94, 96, 149
 set_actionOnClose() (MainWindow method), 133, 138
 set_actions_on_click() (CollectionDisplayer method), 129, 135
 set_actions_on_click() (Widget_graphsVisual method), 95, 96, 138, 149
 set_actionsOnClick() (OptimizationDisplayer method), 130, 136
 set_brush() (TraceVisual method), 92, 145
 set_brushes() (TraceVisual method), 92, 145
 set_collection() (CollectionExporterGUI method), 100
 set_color() (Data method), 34, 54, 140
 set_color() (TraceVisual method), 91, 144
 set_color_palette() (GraphVisual method), 87, 147
 set_convergence() (OptiHistoric method), 78, 85
 set_curr_brush() (_AnimationTrace method), 97
 set_curr_step() (EvolutionaryConvergence method), 67, 69
 set_currFigure() (WindowHolders method), 132, 142
 set_data() (Data method), 32, 52, 138
 set_data() (ListDataStruct method), 20, 29, 50
 set_data_at_index() (ListDataStruct method), 20, 29, 50
 set_data_at_index() (Performance_ListDataStruct method), 30, 51
 set_data_at_indices() (Performance_ListDataStruct method), 30, 51
 set_deviceDrawer() (ContextHandler method), 117
 set_deviceDrawer() (Widget_openGL method),

- 124, 127, 150
- set_deviceToDraw() (*ContextHandler* method), 117
- set_deviceToDraw() (*Widget_openGL* method), 124, 127, 150
- set_entries() (*Widget_tableWithSearch* method), 125, 127, 151
- set_evaluationFunction() (*Monobjective_PSO* method), 73, 77, 83
- set_evaluationFunction() (*MultiObjective_GA* method), 75, 76, 82
- set_evaluationFunction() (*NLOpt_Algorithm* method), 71
- set_filename() (*AutosaveStruct* method), 20, 28, 49
- set_font() (*myLegend* method), 90
- set_fontLabel() (*GraphVisual* method), 87, 146
- set_fontLegend() (*GraphVisual* method), 87, 147
- set_fontTicks() (*GraphVisual* method), 86, 146
- set_graph_disposition() (*myGraphicsLayout* method), 89
- set_graph_disposition() (*Widget_get_graphsVisualLite* method), 93, 95, 148
- set_graph_properties() (*GraphVisual* method), 87, 147
- set_idle_brush() (*_AnimationTrace* method), 97
- set_image() (*Widget_image* method), 122, 126, 149
- set_indices_points_to_plot() (*Data* method), 34, 54, 140
- set_item() (*Widget_tableWithSearch* method), 125, 127, 151
- set_kwargs() (*Data* method), 32, 52, 138
- set_label_pos() (*GraphVisual* method), 87, 147
- set_label_pos() (*myAxis* method), 91
- set_legend() (*Data* method), 34, 54, 140
- set_legend() (*GraphVisual* method), 87, 147
- set_lims() (*GraphVisual* method), 87, 147
- set_lines() (*Widget_lineDrawer* method), 122, 126, 150
- set_list() (*Widget_listWithSearch* method), 123, 127, 149
- set_maxtime() (*Monobjective_PSO* method), 73, 77, 83
- set_maxtime() (*MultiObjective_GA* method), 75, 76, 83
- set_maxtime() (*NLOpt_Algorithm* method), 72
- set_meta() (*Data* method), 32, 52, 138
- set_number_ticks() (*myAxis* method), 91
- set_numberTicks() (*GraphVisual* method), 87, 146
- set_offset() (*myItemSample* method), 90
- set_offset_sample() (*myLegend* method), 90
- set_option() (*Option_class* method), 19, 41, 59
- set_permutations() (*Data* method), 33, 53, 139
- set_pop_size() (*ConvergenceManager* method), 71
- set_position() (*myLegend* method), 90
- set_refreshTime() (*AnimationGUI* method), 98
- set_results() (*OptiHistoric* method), 78, 85
- set_shadow() (*CollectionDisplayer* method), 129, 135
- set_shadow_collection() (*LinkDataGraph* method), 38, 57
- set_space_sample_label() (*myLegend* method), 90
- set_specialButtonsMapping() (*ContextHandler* method), 117
- set_symbol() (*TraceVisual* method), 92, 145
- set_symbolPen() (*TraceVisual* method), 93, 145
- set_symbolPens() (*TraceVisual* method), 93, 146
- set_text() (*Widget_text* method), 126, 128, 151
- set_text() (*Widget_text_scrollable* method), 126, 128, 151
- set_title() (*_PlotHolders* method), 132, 142
- set_title() (*GraphVisual* method), 87, 147
- set_title() (*in module optimeed.visualize*), 142
- set_title() (*in module optimeed.visualize.fastPlot*), 132
- set_title() (*Widget_graphsVisualLite* method), 94, 96, 149
- set_title() (*WindowHolders* method), 132, 142
- set_value() (*Base_Option* method), 40, 58
- set_value() (*Option_bool* method), 41, 58
- set_value() (*Option_dict* method), 41, 59
- set_value() (*Option_float* method), 41, 59
- set_value() (*Option_int* method), 19, 41, 58
- set_value() (*Option_str* method), 41, 58
- set_width_cell() (*myItemSample* method), 90
- set_width_cell_sample() (*myLegend* method), 90
- setCurrentShow() (*in module optimeed.core*), 47
- setCurrentShow() (*in module optimeed.core.commonImport*), 31
- setMaximum() (*widget_doubleSlider* method), 122
- setMinimum() (*widget_doubleSlider* method), 122
- setPath_workspace() (*in module optimeed.core*), 46
- setPath_workspace() (*in module optimeed.core.tools*), 44
- setSingleStep() (*widget_doubleSlider* method), 122
- setText() (*myLabelItem* method), 90
- setValue() (*widget_doubleSlider* method), 122
- SeveralTerminationCondition (*class in optimeed.optimize.optiAlgorithms.multiObjective_GA*), 74
- shouldTerminate() (*ConvergenceTerminationCondition* method), 74
- shouldTerminate() (*MaxTimeTerminationCondition* method), 74
- shouldTerminate() (*SeveralTerminationCondition* method), 74

- method), 74
- show() (in module *optimeed.visualize*), 142
- show() (in module *optimeed.visualize.fastPlot*), 132
- show() (*TraceVisual* method), 92, 145
- show() (*WindowHolders* method), 132, 142
- show_all() (*_AnimationTrace* method), 97
- show_all() (*AnimationGUI* method), 97
- SHOW_CONSTRAINTS (*OptimizationDisplayer* attribute), 130, 136
- SHOW_CURRENT (in module *optimeed.core*), 47
- SHOW_CURRENT (in module *optimeed.core.commonImport*), 31
- SHOW_DEBUG (in module *optimeed.core*), 47, 48
- SHOW_DEBUG (in module *optimeed.core.commonImport*), 31
- SHOW_ERROR (in module *optimeed.core*), 47, 48, 60
- SHOW_ERROR (in module *optimeed.core.commonImport*), 31
- SHOW_INFO (in module *optimeed.core*), 47, 48
- SHOW_INFO (in module *optimeed.core.commonImport*), 31
- SHOW_LOGS (in module *optimeed.core*), 47
- SHOW_LOGS (in module *optimeed.core.commonImport*), 31
- SHOW_WARNING (in module *optimeed.core*), 47, 48, 51, 60, 62
- SHOW_WARNING (in module *optimeed.core.commonImport*), 31
- SHOW_WARNING (in module *optimeed.visualize*), 143
- showEvent() (*Widget_menuButton* method), 124, 127, 148, 150
- showRow() (*Widget_tableWithSearch* method), 125, 127, 150
- signal_graph_changed (*Widget_graphsVisualLite* attribute), 93, 95, 148
- signal_has_exported (*CollectionExporterGUI* attribute), 99
- signal_has_reset (*CollectionExporterGUI* attribute), 100
- signal_must_update (*TraceVisual* attribute), 91, 144
- signal_must_update (*Widget_graphsVisualLite* attribute), 93, 95, 148
- signal_must_update (*Widget_lineDrawer* attribute), 122, 126, 150
- signal_optimization_over (*OptimizationDisplayer* attribute), 130, 136
- Silver_material (in module *optimeed.visualize*), 151
- Silver_material (in module *optimeed.visualize.widgets*), 128
- Silver_material (in module *optimeed.visualize.widgets.openGL*), 121
- Silver_material (in module *optimeed.visualize.widgets.openGL.materials*), 119
- SingleObjectSaveLoad (class in *optimeed.core*), 48
- SingleObjectSaveLoad (class in *optimeed.core.collection*), 28
- singleStep() (*widget_doubleSlider* method), 122
- sizeHint() (*Widget_openGL* method), 124, 127, 150
- slider_handler() (*AnimationGUI* method), 97
- SLIDER_MAXIMUM_VALUE (*AnimationGUI* attribute), 97
- SLIDER_MINIMUM_VALUE (*AnimationGUI* attribute), 97
- software_version() (in module *optimeed.core*), 45
- software_version() (in module *optimeed.core.tools*), 43
- sortByDimension() (*HyperVolume* method), 68
- sparse_subset() (in module *optimeed.core*), 60
- sparse_subset() (in module *optimeed.core.additional_tools*), 27
- SpecialButtonsMapping (class in *optimeed.visualize.widgets.openGL.contextHandler*), 117
- start() (*_Evaluator* method), 80
- start() (*OptiHistoric* method), 78, 86
- start_autorefresh() (*OptimizationDisplayer* method), 130, 137
- start_autosave() (*AutosaveStruct* method), 21, 28, 49
- start_qt_mainloop() (in module *optimeed.visualize*), 138
- start_qt_mainloop() (in module *optimeed.visualize.process_mainloop*), 133
- Steel_material (in module *optimeed.visualize*), 152
- Steel_material (in module *optimeed.visualize.widgets*), 128
- Steel_material (in module *optimeed.visualize.widgets.openGL*), 121
- Steel_material (in module *optimeed.visualize.widgets.openGL.materials*), 119
- stop_autorefresh() (*OptimizationDisplayer* method), 130, 137
- stop_autosave() (*AutosaveStruct* method), 20, 28, 49
- stop_qt_mainloop() (in module *optimeed.visualize.process_mainloop*), 134
- str_all_attr() (in module *optimeed.core*), 46
- str_all_attr() (in module *optimeed.core.tools*), 44
- success (*SensitivityResults* attribute), 17
- SurfPlot (class in *optimeed.core*), 56
- SurfPlot (class in *optimeed.core.graphs3*), 36
- SurfPlot (class in *optimeed.visualize*), 143
- symbol_isfilled() (*Data* method), 32, 52, 138

T

templates_tikz (in module *optimeed.core.tikzTranslator*), 42
 text_format (class in *optimeed.core*), 45
 text_format (class in *optimeed.core.tools*), 43
 theLock (in module *optimeed.core*), 50
 theLock (in module *optimeed.core.collection*), 30
 time (*OptiHistoric.pointData* attribute), 77, 85
 to_css_classes() (*_State* method), 24
 toggle() (*TraceVisual* method), 92, 145
 toolTip() (*Plugin_listWithSearch* method), 123
 toolTip() (*Plugin_tableWithSearch* method), 125
 TraceVisual (class in *optimeed.visualize*), 144
 TraceVisual (class in *optimeed.visualize.graphs.traceVisual*), 91
 TraceVisual._ModifiedPaintElem (class in *optimeed.visualize*), 144
 TraceVisual._ModifiedPaintElem (class in *optimeed.visualize.graphs.traceVisual*), 91
 truncate() (in module *optimeed.core*), 46
 truncate() (in module *optimeed.core.tools*), 44

U

UNDERLINE (*text_format* attribute), 43, 45
 universalPath() (in module *optimeed.core*), 46
 universalPath() (in module *optimeed.core.tools*), 44
 update() (*GraphVisual* method), 88, 147
 update_graphs() (*CollectionDisplay* method), 129, 135
 update_graphs() (*LinkDataGraph* method), 38, 57
 update_graphs() (*Widget_graphsVisualLite* method), 94, 95, 148
 update_label (*myAxis* attribute), 90
 update_widget_w_animation() (*Animate_lines* method), 99, 112, 157
 update_widget_w_animation() (*Animate_lines_and_text* method), 99, 113, 158
 update_widget_w_animation() (*Animate_openGL* method), 98, 113, 157
 update_widget_w_animation() (*Animate_openGL_and_text* method), 98, 113, 158
 updateChildren() (*Graphs* method), 34, 54, 140
 updateSize() (*myLegend* method), 90
 updateTrace() (*TraceVisual* method), 92, 145
 useOpenGL() (*myGraphicsLayoutWidget* method), 89, 144

V

val_max (*Integer_OptimizationVariable* attribute), 79, 84
 val_max (*Real_OptimizationVariable* attribute), 79, 84

val_min (*Integer_OptimizationVariable* attribute), 79, 84
 val_min (*Real_OptimizationVariable* attribute), 79, 84
 value (*Option_bool* attribute), 41, 58
 value (*Option_dict* attribute), 41, 59
 value (*Option_float* attribute), 41, 59
 value (*Option_int* attribute), 19, 41, 58
 value (*Option_str* attribute), 41, 58
 value() (*widget_doubleSlider* method), 122
 VERSION (in module *optimeed*), 159
 ViewOptimizationResults (class in *optimeed.visualize*), 137
 ViewOptimizationResults (class in *optimeed.visualize.viewOptimizationResults*), 134
 VT100_BOX_CODES (in module *optimeed.core.ansi2html.converter*), 24

W

whatsThis() (*Plugin_listWithSearch* method), 123
 whatsThis() (*Plugin_tableWithSearch* method), 125
 wheelEvent() (*Widget_openGL* method), 124, 127, 150
 WHITE (*text_format* attribute), 43, 45
 widget_doubleSlider (class in *optimeed.visualize.widgets.widget_doubleSlider*), 122
 Widget_graphsVisual (class in *optimeed.visualize*), 137, 149
 Widget_graphsVisual (class in *optimeed.visualize.graphs*), 96
 Widget_graphsVisual (class in *optimeed.visualize.graphs.widget_graphsVisual*), 94
 Widget_graphsVisualLite (class in *optimeed.visualize*), 148
 Widget_graphsVisualLite (class in *optimeed.visualize.graphs*), 95
 Widget_graphsVisualLite (class in *optimeed.visualize.graphs.widget_graphsVisual*), 93
 Widget_image (class in *optimeed.visualize*), 149
 Widget_image (class in *optimeed.visualize.widgets*), 126
 Widget_image (class in *optimeed.visualize.widgets.widget_image*), 122
 Widget_lineDrawer (class in *optimeed.visualize*), 149
 Widget_lineDrawer (class in *optimeed.visualize.widgets*), 126
 Widget_lineDrawer (class in *optimeed.visualize.widgets.widget_lineDrawer*), 122

Widget_listWithSearch (class in *optimeed.visualize*), 149

Widget_listWithSearch (class in *optimeed.visualize.widgets*), 126

Widget_listWithSearch (class in *optimeed.visualize.widgets.widget_listWithSearch*), 123

Widget_menuButton (class in *optimeed.visualize*), 147, 150

Widget_menuButton (class in *optimeed.visualize.widgets*), 127

Widget_menuButton (class in *optimeed.visualize.widgets.widget_menuButton*), 124

Widget_openGL (class in *optimeed.visualize*), 150

Widget_openGL (class in *optimeed.visualize.widgets*), 127

Widget_openGL (class in *optimeed.visualize.widgets.widget_openGL*), 124

Widget_tableWithSearch (class in *optimeed.visualize*), 150

Widget_tableWithSearch (class in *optimeed.visualize.widgets*), 127

Widget_tableWithSearch (class in *optimeed.visualize.widgets.widget_tableWithSearch*), 125

Widget_text (class in *optimeed.visualize*), 151

Widget_text (class in *optimeed.visualize.widgets*), 128

Widget_text (class in *optimeed.visualize.widgets.widget_text*), 126

Widget_text_scrollable (class in *optimeed.visualize*), 151

Widget_text_scrollable (class in *optimeed.visualize.widgets*), 128

Widget_text_scrollable (class in *optimeed.visualize.widgets.widget_text*), 126

WindowHolders (class in *optimeed.visualize*), 142

WindowHolders (class in *optimeed.visualize.fastPlot*), 132

Y

YELLOW (*text_format* attribute), 43, 45

Yellow_Emerald_material (in module *optimeed.visualize*), 151

Yellow_Emerald_material (in module *optimeed.visualize.widgets*), 128

Yellow_Emerald_material (in module *optimeed.visualize.widgets.openGL*), 121

Yellow_Emerald_material (in module *optimeed.visualize.widgets.openGL.materials*), 119