

---

# **optimeed**

***Release 1.0***

**Jan 06, 2020**



---

## Contents

---

<b>1</b>	<b>Requirements</b>	<b>3</b>
<b>2</b>	<b>Installation</b>	<b>5</b>
<b>3</b>	<b>Quickstart</b>	<b>7</b>
3.1	Quickstart Optimization . . . . .	7
3.2	Quickstart Visualization . . . . .	9
<b>4</b>	<b>Gallery</b>	<b>13</b>
4.1	Gallery . . . . .	13
<b>5</b>	<b>License and support</b>	<b>15</b>
5.1	License and Support . . . . .	15
<b>6</b>	<b>API</b>	<b>17</b>
6.1	:mod:optimeed . . . . .	17
<b>7</b>	<b>Developer guide</b>	<b>111</b>
7.1	documentation . . . . .	111
	<b>Python Module Index</b>	<b>113</b>
	<b>Index</b>	<b>115</b>



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



# CHAPTER 1

---

## Requirements

---

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





## CHAPTER 2

---

### Installation

---

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

```
pip install optimeed
```

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

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



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

## 3.1 Quickstart Optimization

An optimization process can be presented as following:

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

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

To set up optimization project, begin with these imports

```
from optimeed.core import InterfaceDevice
from optimeed.optimize.optiAlgorithms import MultiObjective_GA as _
↳ OptimizationAlgorithm
# from optimeed.optimize.optiAlgorithms import NLOpt_Algorithm as _
↳ OptimizationAlgorithm
```

(continues on next page)

(continued from previous page)

```

from optimeed.optimize import Optimizer, Real_OptimizationVariable, FastObjCons, ↪
↪InterfaceCharacterization
from optimeed.visualize import start_qt_mainloop
import time

```

Then define the device to optimize

```

class Device(InterfaceDevice):
    def __init__(self):
        self.x = 1
        self.y = 1

```

Define how the device will be characterized. In this example nothing happens (but a sleep). This step is not mandatory

```

class Characterization(InterfaceCharacterization):
    def compute(self, theDevice):
        time.sleep(0.005)

```

Once the classes are defined, we can start to instantiate them:

```

theDevice = Device()
theMathsToPhysics = MathsToPhysics()
theAlgo = OptimizationAlgorithm()
theCharacterization = Characterization()

```

The optimization algorithm supports multicore usage (as option, default to be one):

```

theAlgo.set_optionValue(theAlgo.NUMBER_OF_CORES, 4)

```

We then set the variables to be optimized:

```

optimizationVariables = list()
optimizationVariables.append(Real_OptimizationVariable('x', 0, 2))
optimizationVariables.append(Real_OptimizationVariable('y', 0, 2))

```

And the objectives and constraints:

```

listOfObjectives = [FastObjCons("( 1 + ({y}+3)*{x}**2 )**0.5"), FastObjCons("4 + 2*(
↪{y}+3)**0.5*(1+({x}-1)**2)**0.5 ")]
listOfConstraints = [FastObjCons("{x} - 0.55")]

```

Finally set the optimizer:

```

theOptimizer = Optimizer()
theOptimizer.set_optionValue(theOptimizer.KWARGS_OPTIHISTO, {"autosave": False})
PipeOptimization = theOptimizer.set_optimizer(theDevice, listOfObjectives, ↪
↪listOfConstraints, optimizationVariables,
theOptimizationAlgorithm=theAlgo, ↪
↪theCharacterization=theCharacterization)
theOptimizer.set_max_opti_time(2)

```

The optimizer can then be run with:

```

result, convergence = theOptimizer.run_optimization()

```

Or, if visualisation is needed:

```
from optimeed.visualize.displayOptimization import OptimizationDisplay
optiDisplayer = OptimizationDisplay(PipeOptimization, listOfObjectives,
    ↪theOptimizer)
_, _ = optiDisplayer.generate_optimizationGraphs()
resultsOpti, convergence = optiDisplayer.launch_optimization()
```

Finally:

```
print("Best individuals :")
for device in resultsOpti:
    print("x : {} \t y : {}".format(device.x, device.y))
```

## 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:

This example shows how to create a simple gui that contains an openGL widget. First define the imports:

```
from optimeed.visualize.gui.widgets.widget_opengl import widget_opengl
from optimeed.visualize.gui.gui_mainWindow import gui_mainWindow

from optimeed.visualize.gui.widgets.openglWidget.DeviceDrawerInterface import
    ↪DeviceDrawerInterface
from optimeed.core.interfaceDevice import InterfaceDevice
from optimeed.visualize.gui.widgets.openglWidget.OpenGLFunctions_Library import *
from optimeed.visualize.gui.widgets.openglWidget.Materials_visual import *
```

Define the device to draw

```
class Cone(InterfaceDevice):
    def __init__(self):
        self.radius_base = 1
        self.height = 1.5
```

Define the drawer

```
class ConeDrawer(DeviceDrawerInterface):
    def __init__(self):
        self.theCone = None

    def draw(self, theCone): # How to draw the cone
        self.theCone = theCone
        glPushMatrix() # Remove the previous matrices transformations
        glTranslate(0, 0, -theCone.height/2) # Move the cone
```

(continues on next page)

(continued from previous page)

```

        Bronze_material.activateMaterialProperties() # Change colour aspect of the_
↪cones
        draw_disk(0, theCone.radius_base, 50, translate=theCone.height) # Draw the_
↪base
        gluCylinder(gluNewQuadric(), 0, theCone.radius_base, theCone.height, 50, 10)
↪# Draw the cylinder
        glPopMatrix() # Push back previous matrices transformations

    def get_init_camera(self, theDevice):
        tipAngle = 10
        viewAngle = 10
        zoomLevel = 0.5
        return tipAngle, viewAngle, zoomLevel

    def keyboard_push_action(self, theKey):
        if theKey == ord(b'H'):
            self.theCone.radius_base += 0.2 # Change the radius length when h is_
↪pressed

```

Instantiates objects and run the code

```

openGlWidget = widget_openGL()
theDrawer = ConeDrawer()
theCone = Cone()
openGlWidget.set_deviceDrawer(theDrawer)
openGlWidget.set_deviceToDraw(theCone)
myWindow = gui_mainWindow([openGlWidget], keep_alive=True)
myWindow.run()

```

### 3.2.2 Advanced visualization:

This example truly shows the potential of this tool, by linking saved data to graphs.

First, define the imports:

```

from optimeed.core import Collection
# Visuals imports
from optimeed.core.linkDataGraph import LinkDataGraph, HowToPlotGraph
from optimeed.visualize.gui.gui_mainWindow import gui_mainWindow
# Graph visuals imports
from optimeed.visualize.gui.widgets.widget_graphs_visual import widget_graphs_visual
from optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick import *
from optimeed.visualize.gui.widgets.graphsVisualWidget.smallGui import guiPyqtgraph
# OpenGL imports
from optimeed.visualize.gui.widgets.widget_openGL import widget_openGL
from optimeed.visualize.gui.widgets.openGLWidget.DeviceDrawerInterface import_
↪DeviceDrawerInterface
from optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctions_Library import *
from optimeed.visualize.gui.widgets.openGLWidget.Materials_visual import *

import os

```

Then, define an openGL drawer:

```

class Drawer(DeviceDrawerInterface):
    def __init__(self):
        self.theDevice = None

    def draw(self, theDevice):
        self.theDevice = theDevice
        glPushMatrix()
        Bronze_material.activateMaterialProperties()
        draw_simple_rectangle(theDevice.x, theDevice.y)
        glPopMatrix()

    def get_init_camera(self, theDevice):
        return 0, 0, 0.5

```

Load data files. Path is relative to this directory `__file__`:

```

collection_devices = Collection.load(os.path.join(os.path.dirname(__file__),
↪ 'resources/autosaved.col'), doCoherence=False)
collection_logOpti = Collection.load(os.path.join(os.path.dirname(__file__),
↪ 'resources/logopti.col'), doCoherence=False)

```

Instantiates high level module that links the data contained in collections to graphs (that will be later created):

```

theDataLink = LinkDataGraph()
id_logOpti = theDataLink.add_collection(collection_logOpti)
id_devices = theDataLink.add_collection(collection_devices)

```

The attributes to plots on x and y axis, and additional kwargs.:

```

howToPlot = HowToPlotGraph('objectives[0]', 'objectives[1]', {'x_label': "Objective 1
↪", 'y_label': "Objective 2", 'is_scattered': True})

```

The trick here is that the objective functions is not directly stocked in `collection_devices` but in `collection_logOpti`. So we display the objectives coming from `collection_logOpti` but we link `collection_devices` from it:

```

howToPlot.exclude_col(id_devices)
theDataLink.link_collection_to_graph_collection(id_logOpti, id_devices)  # Link the_
↪ devices to the logopti

```

Generate the graphs:

```

theDataLink.add_graph(howToPlot)
theGraphs = theDataLink.createGraphs()

```

Add additional actions to perform when the graph is clicked. This is what makes this software extremely powerful.:

```

theActionsOnClick = list()

openGLDrawing = widget_openGL()
openGLDrawing.set_deviceDrawer(Drawer())

theActionsOnClick.append(on_graph_click_showAnim(theDataLink, _
↪ DataAnimationOpenGL(openGLDrawing)))
theActionsOnClick.append(on_graph_click_showInfo(theDataLink, visuals=[Repr_
↪ opengl(Drawer())]))
theActionsOnClick.append(on_click_extract_pareto(theDataLink, max_x=False, max_
↪ y=False))
theActionsOnClick.append(on_graph_click_delete(theDataLink))

```

Create the widget of the graphs, and the associated GUI:

```
myWidgetGraphsVisuals = widget_graphs_visual(theGraphs, highlight_last=True, refresh_  
↪time=-1)  
guiPyqtgraph(myWidgetGraphsVisuals, actionsOnClick=theActionsOnClick) # Add GUI to_  
↪change action easily and export graphs  
myWidgetGraphsVisuals = myWidgetGraphsVisuals
```

Launch the window:

```
myWindow = gui_mainWindow([myWidgetGraphsVisuals], keep_alive=True)  
myWindow.run()
```



## 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.degreeef@uclouvain.be](mailto:christophe.degreeef@uclouvain.be)



## 6.1 :mod:optimeed

### 6.1.1 Subpackages

`consolidate`

`parametric_analysis`

#### Module Contents

```

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

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

    get_reference_device (self)

    get_analyzed_attribute (self)

class Parametric_minmax (analyzed_attribute, reference_device, min_value, max_value, is_adim=False,
                           npoints=10)
    Bases: optimeed.consolidate.parametric_analysis.Parametric_parameter

    get_values (self)

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

    NUMBER_OF_CORES = 1

    run (self)
        Instantiates input arguments for analysis

```

```
evaluate (self, theDevice)
initialize_output_collection (self)
```

## Package Contents

```
class Option_class
```

```
    get_optionValue (self, optionId)
    set_optionValue (self, optionId, value)
    get_all_options (self)
    set_all_options (self, options)
    add_option (self, idOption, name, value)
```

```
getPath_workspace ()
```

```
rsetattr (obj, attr, val)
```

```
rgetattr (obj, attr)
```

Recursively get an attribute from object. Extends getattr method

### Parameters

- **obj** – object
- **attr** – attribute to get

### Returns

```
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
```

```
indentParagraph (text_in, indent_level=1)
```

```
class Parametric_Collection (**kwargs)
```

Bases: optimeed.core.collection.Collection

```
class Parametric_parameter (analyzed_attribute, reference_device)
```

Abstract class for a parametric parameter

```

    get_reference_device (self)
    get_analyzed_attribute (self)
class Parametric_minmax (analyzed_attribute, reference_device, min_value, max_value, is_adim=False,
                          npoints=10)
    Bases: optimeed.consolidate.parametric_analysis.Parametric_parameter
    get_values (self)
class Parametric_analysis (theParametricParameter,          theCharacterization,          file-
                          name_collection=None,              description_collection=None,          au-
                          tosave=False)
    Bases: optimeed.core.Option_class
    NUMBER_OF_CORES = 1
    run (self)
        Instantiates input arguments for analysis
    evaluate (self, theDevice)
    initialize_output_collection (self)

```

## core

### Subpackages

ansi2html

converter

### Module Contents

```

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 = \documentclass{scrartcl}

usepackage[utf8]{inputenc}      usepackage{fancyvrb}      usepackage[usenames,dvipsnames]{xcolor}      %%
definecolor{red-sd}{HTML}{7ed2d2}

title{%(title)s}

fvset{commandchars=\{\}}

begin{document}

begin{Verbatim} %(content)s end{Verbatim} end{document}

_html_template

class _State
    Bases: object

    reset (self)

    adjust (self, ansi_code, parameter=None)

    to_css_classes (self)

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** (*self*, *ansi*)

**\_apply\_regex** (*self*, *ansi*, *styles\_used*)

**\_collapse\_cursor** (*self*, *parts*)

Act on any CursorMoveUp commands by deleting preceding tokens

**prepare** (*self*, *ansi*="", *ensure\_trailing\_newline*=False)

Load the contents of 'ansi' into this object

**attrs** (*self*)

Prepare attributes for the template

**convert** (*self*, *ansi*, *full*=True, *ensure\_trailing\_newline*=False)

**produce\_headers** (*self*)

**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

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

Bases: object

**\_\_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

**read\_to\_unicode** (*obj*)

## Package Contents

```
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** (*self, ansi*)

**\_apply\_regex** (*self, ansi, styles\_used*)

**\_collapse\_cursor** (*self, parts*)

Act on any CursorMoveUp commands by deleting preceding tokens

**prepare** (*self, ansi="", ensure\_trailing\_newline=False*)

Load the contents of 'ansi' into this object

**attrs** (*self*)

Prepare attributes for the template

**convert** (*self, ansi, full=True, ensure\_trailing\_newline=False*)

**produce\_headers** (*self*)

## collection

## Module Contents

```
class DataStruct_Interface
```

**get\_info** (*self*)

Get simple string describing the datastructure

**set\_info** (*self, info*)

Set simple string describing the datastructure

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

```
class AutosaveStruct (dataStruct, filename="", change_filename_if_exists=True)
```

Structure that provides automated save of DataStructures

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

**get\_filename** (*self*)

Get set filename

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

### Parameters

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

**stop\_autosave** (*self*)

Stop autosave

```

start_autosave (self, timer_autosave)
    Start autosave

save (self, safe_save=True)
    Save

get_datastruct (self)
    Return :class:'~DataStruct_Interface'

class ListDataStruct
    Bases: optimeed.core.collection.DataStruct_Interface
    _INFO_STR = info
    _DATA_STR = data

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

    add_data (self, data_in)
        Add a data to the list

    get_data (self)
        Get full list of datas

    set_data (self, theData)
        Set full list of datas

    set_data_at_index (self, data_in, index)
        Replace data at specific index

    set_attribute_data (self, the_attribute, the_value)
        Set attribute to all data

    set_attribute_equation (self, attribute_name, equation_str)
        Advanced method to set the value of attribute_name from equation_str

        Parameters

- attribute_name – string (name of the attribute to set)
- equation_str – formatted equation, check applyEquation()

Returns
        list

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

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

        Returns list

    delete_points_at_indices (self, indices)
        Delete several elements from the Collection

        Parameters indices – list of indices to delete

    export_xls (self, excelFilename, excelsheet='Sheet1', mode='w')
        Export the collection to excel. It only exports the direct attributes.

        Parameters

- excelFilename – filename of the excel
- excelsheet – name of the sheet
- mode – 'w' to erase existing file, 'a' to append sheetname to existing file

```

**merge** (*self*, *collection*)

Merge a collection with the current collection

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

**color\_palette**

## Module Contents

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

**blackOnly** (*N*)

**dark2** (*N*)

**commonImport**

## Module Contents

**SHOW\_WARNING** = 0

**SHOW\_INFO** = 1

**SHOW\_ERROR** = 2

**SHOW\_DEBUG** = 3

**SHOW\_CURRENT**

**graphs**

## Module Contents

**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*, *symbol='o'*, *symbolsize=8*, *fillsymbol=True*, *outlinesymbol=1.8*, *linestyle='-'*, *width=2*)

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

Overwrites current datapoints with new set

**get\_x** (*self*)

Get x coordinates of datapoints

**get\_symbolsize** (*self*)

Get size of the symbols

**symbol\_isfilled** (*self*)

Check if symbols has to be filled or not

**get\_symbolOutline** (*self*)

Get color factor of outline of symbols

**get\_length\_data** (*self*)

Get number of points

**get\_xlim**(*self*)  
Get x limits of viewbox

**get\_ylim**(*self*)  
Get y limits of viewbox

**get\_y**(*self*)  
Get y coordinates of datapoints

**get\_color**(*self*)  
Get color of the line

**get\_width**(*self*)  
Get width of the line

**get\_number\_of\_points**(*self*)  
Get number of points

**get\_plot\_data**(*self*)  
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\_permutations**(*self*)  
Return the transformation 'permutation': `xplot[i] = xdata[permutation[i]]`

**get\_invert\_permutations**(*self*)  
Return the inverse of permutations: `xdata[i] = xplot[revert[i]]`

**get\_dataIndex\_from\_graphIndex**(*self*, *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**(*self*, *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**(*self*, *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**(*self*, *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**(*self*, *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** (*self*)  
Get x label of the trace

**get\_y\_label** (*self*)  
Get y label of the trace

**get\_legend** (*self*)  
Get name of the trace

**get\_symbol** (*self*)  
Get symbol

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

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

**is\_scattered** (*self*)  
Delete a point from the datapoints

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

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

**get\_linestyle** (*self*)  
Get linestyle

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

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

**class Graph**

Simple graph container that contains several traces

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

**Parameters** *data* – *Data*

**Returns** id of the created trace

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

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

**get\_trace** (*self*, *idTrace*)  
Get data object of *idTrace*

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

**Returns** *Data*

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

**export\_str** (*self*)

**class Graphs**  
Contains several *Graph*

**updateChildren** (*self*)

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

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

**add\_trace** (*self*, *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** (*self*, *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** (*self*)

Get id of the first graph

**Returns** id of the first graph

**get\_graph** (*self*, *idGraph*)

Get graph object at idgraph

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

**Returns** *Graph*

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

Get all ids of the graphs

**Returns** list of id graphs

**get\_all\_graphs** (*self*)

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

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

Add a new graph

**Returns** id of the created graph

**remove\_graph** (*self*, *idGraph*)

Delete a graph

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

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

Add a callback each time a graph is modified.

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

**export\_str** (*self*)

Export all the graphs in text

**Returns** str

**merge** (*self*, *otherGraphs*)

```
reset (self)
```

```
interfaceDevice
```

## Module Contents

```
class InterfaceDevice
```

Interface class that represents a device. Hidden feature: variables that need to be saved must be type-hinted: e.g.: `x: int`. See `obj_to_json()` for more info

```
assign (self, machine_to_assign, resetAttribute=False)
```

Copy the attribute values of machine\_to\_assign to self. The references are not lost.

### Parameters

- `machine_to_assign` – InterfaceDevice
- `resetAttribute` –

```
linkDataGraph
```

## Module Contents

```
class HowToPlotGraph (attribute_x, attribute_y, kwargs_graph=None, excluded=None)
```

```
exclude_col (self, id_col)
```

Add id\_col to exclude from the graph

```
__str__ (self)
```

```
class CollectionInfo (theCollection, kwargs, theID)
```

```
get_collection (self)
```

```
get_kwargs (self)
```

```
get_id (self)
```

```
class LinkDataGraph
```

```
class _collection_linker
```

```
add_link (self, idSlave, idMaster)
```

```
get_collection_master (self, idToGet)
```

```
is_slave (self, idToCheck)
```

```
set_same_master (self, idExistingSlave, idOtherSlave)
```

### Parameters

- `idExistingSlave` – id collection of the existing slave
- `idOtherSlave` – id collection of the new slave that has to be linked to an existing master

```
add_collection (self, theCollection, kwargs=None)
```



```

add_graph (self, howToPlotGraph)
createGraphs (self)
get_howToPlotGraph (self, idGraph)
get_collectionInfo (self, idCollectionInfo)
create_trace (self, collectionInfo, howToPlotGraph, idGraph)
get_all_id_graphs (self)
get_all_traces_id_graph (self, idGraph)
update_graphs (self)
is_slave (self, idGraph, idTrace)
get_idCollection_from_graph (self, idGraph, idTrace, getMaster=True)
    From indices in the graph, get index of corresponding collection
get_collection_from_graph (self, idGraph, idTrace, getMaster=True)
    From indices in the graph, get corresponding collection
get_dataObject_from_graph (self, idGraph, idTrace, idPoint)
get_dataObjects_from_graph (self, idGraph, idTrace, idPoint_list)
remove_element_from_graph (self, idGraph, idTrace, idPoint, deleteFromMaster=False)
    Remove element from the graph, or the master collection
remove_elements_from_trace (self, idGraph, idTrace, idPoints, deleteFromMaster=False)
    Performances      optimisation      when      compared      to      LinkDataGraph.
    remove\_element\_from\_graph\(\)
link_collection_to_graph_collection (self, id_collection_graph, id_collection_master)
    Link data :param id_collection_graph: :param id_collection_master: :return:
remove_trace (self, idGraph, idTrace)
get_graph_and_trace_from_collection (self, idCollection)
    Reverse search: from a collection, get the associated graph
get_mappingData_graph (self, idGraph)
get_mappingData_trace (self, idGraph, idTrace)

```

myjson

## Module Contents

```

MODULE_TAG = __module__
CLASS_TAG = __class__
EXCLUDED_TAGS
_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 :param json\_dict: dictionary loaded from a json file. :raise TypeError: if class can not be found :raise KeyError: if CLASS\_TAG not present in dictionary

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

Safe class to create object from dictionary. :param json\_dict: dictionary loaded from a json file :param cls: class object to instantiate with dictionary

**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.

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

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

**class Bar** (*num*)

*value* :int

**options**

## Module Contents

**class Options**

**get\_name** (*self, idOption*)

**get\_value** (*self, idOption*)

**add\_option** (*self, idOption, name, value*)

**set\_option** (*self, idOption, value*)

**copy** (*self*)

**set\_self** (*self, the\_options*)

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

**class Option\_class**

**get\_optionValue** (*self, optionId*)

**set\_optionValue** (*self, optionId, value*)

**get\_all\_options** (*self*)

**set\_all\_options** (*self, options*)

**add\_option** (*self, idOption, name, value*)

**tools**

## Module Contents

**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*)

**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** ()

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

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

**universalPath** (*thePath*)

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

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

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

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

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

**quicksort** (*array*)

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

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

Recursively get an attribute from object. Extends getattr method

**Parameters**

- **obj** – object
- **attr** – attribute to get

**Returns**

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

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

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

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

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

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

**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.

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

**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** (*self, 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

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

Delete elements from list at indices :param indices: list :param theList: list

## Package Contents

**getPath\_workspace** ()

**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.

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

Convenience class to create object from dictionary. Only works if CLASS\_TAG is valid :param json\_dict: dictionary loaded from a json file. :raise TypeError: if class can not be found :raise KeyError: if CLASS\_TAG not present in dictionary

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

Safe class to create object from dictionary. :param json\_dict: dictionary loaded from a json file :param cls: class object to instantiate with dictionary

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

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

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

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

Recursively get an attribute from object. Extends getattr method

### Parameters

- **obj** – object
- **attr** – attribute to get

### Returns

**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)

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

**SHOW\_WARNING** = 0

```
class DataStruct_Interface
```

```
    get_info (self)
        Get simple string describing the datastructure

    set_info (self, info)
        Set simple string describing the datastructure

    __str__ (self)
```

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

```
    __str__ (self)

    get_filename (self)
        Get set filename

    set_filename (self, filename, change_filename_if_exists)
```

**Parameters**

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

```
    stop_autosave (self)
        Stop autosave

    start_autosave (self, timer_autosave)
        Start autosave

    save (self, safe_save=True)
        Save

    get_datastruct (self)
        Return :class:`~DataStruct_Interface`
```

```
class ListDataStruct
```

```
    Bases: optimeed.core.collection.DataStruct_Interface
```

```
    _INFO_STR = info

    _DATA_STR = data

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

    add_data (self, data_in)
        Add a data to the list

    get_data (self)
        Get full list of datas

    set_data (self, theData)
        Set full list of datas

    set_data_at_index (self, data_in, index)
        Replace data at specific index

    set_attribute_data (self, the_attribute, the_value)
        Set attribute to all data

    set_attribute_equation (self, attribute_name, equation_str)
        Advanced method to set the value of attribute_name from equation_str
```

**Parameters**

- **attribute\_name** – string (name of the attribute to set)
- **equation\_str** – formatted equation, check `applyEquation()`

**Returns**

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

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

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

**Returns** list

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

Delete several elements from the Collection

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

**export\_xls** (*self, excelFilename, excelsheet='Sheet1', mode='w'*)

Export the collection to excel. It only exports the direct attributes.

**Parameters**

- **excelFilename** – filename of the excel
- **excelsheet** – name of the sheet
- **mode** – 'w' to erase existing file, 'a' to append sheetname to existing file

**merge** (*self, collection*)

Merge a collection with the current collection

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

**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*)

**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** ()

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

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

**universalPath** (*thePath*)

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

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

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

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

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

**quicksort** (*array*)

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

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

Recursively get an attribute from object. Extends getattr method

**Parameters**

- **obj** – object
- **attr** – attribute to get

**Returns**

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

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

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

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

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

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

**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.

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

**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** (*self, 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

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

Delete elements from list at indices :param indices: list :param theList: list

**SHOW\_WARNING** = 0

**SHOW\_INFO** = 1

**SHOW\_ERROR** = 2

**SHOW\_DEBUG** = 3

**SHOW\_CURRENT**

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

**SHOW\_WARNING** = 0

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

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

Overwrites current datapoints with new set

**get\_x** (*self*)

Get x coordinates of datapoints

**get\_symbols\_size** (*self*)

Get size of the symbols

**symbol\_isfilled** (*self*)

Check if symbols has to be filled or not

**get\_symbolOutline** (*self*)

Get color factor of outline of symbols

**get\_length\_data** (*self*)

Get number of points

**get\_xlim** (*self*)

Get x limits of viewbox

**get\_ylim** (*self*)

Get y limits of viewbox

**get\_y** (*self*)

Get y coordinates of datapoints

**get\_color** (*self*)

Get color of the line

**get\_width** (*self*)

Get width of the line

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

Get number of points

**get\_plot\_data** (*self*)

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\_permutations** (*self*)

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

**get\_invert\_permutations** (*self*)

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

**get\_dataIndex\_from\_graphIndex** (*self*, *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** (*self*, *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** (*self*, *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** (*self*, *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** (*self*, *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** (*self*)

Get x label of the trace

**get\_y\_label** (*self*)

Get y label of the trace

**get\_legend** (*self*)

Get name of the trace

**get\_symbol** (*self*)

Get symbol

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

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

**delete\_point** (*self*, *index\_point*)

Delete a point from the datapoints

**is\_scattered** (*self*)

Delete a point from the datapoints

**set\_indices\_points\_to\_plot** (*self*, *indices*)

Set indices points to plot

**get\_indices\_points\_to\_plot** (*self*)

Get indices points to plot

**get\_linestyle** (*self*)

Get linestyle

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

**export\_str** (*self*)

Method to save the points constituting the trace

**class Graph**

Simple graph container that contains several traces

**add\_trace** (*self*, *data*)

Add a trace to the graph

**Parameters** `data` – *Data*

**Returns** id of the created trace

**remove\_trace** (*self*, *idTrace*)

Delete a trace from the graph

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

**get\_trace** (*self*, *idTrace*)  
Get data object of idTrace

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

**Returns** *Data*

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

**export\_str** (*self*)

**class** **Graphs**

Contains several *Graph*

**updateChildren** (*self*)

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

**add\_trace** (*self*, *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** (*self*, *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** (*self*)  
Get id of the first graph

**Returns** id of the first graph

**get\_graph** (*self*, *idGraph*)  
Get graph object at idgraph

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

**Returns** *Graph*

**get\_all\_graphs\_ids** (*self*)  
Get all ids of the graphs

**Returns** list of id graphs

**get\_all\_graphs** (*self*)  
Get all graphs. Return dict {id: *Graph*}

**add\_graph** (*self*, *updateChildren=True*)  
Add a new graph

**Returns** id of the created graph

**remove\_graph** (*self*, *idGraph*)  
Delete a graph

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

**add\_update\_method** (*self*, *childObject*)  
Add a callback each time a graph is modified.

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

**export\_str** (*self*)  
Export all the graphs in text

**Returns** str

**merge** (*self*, *otherGraphs*)

**reset** (*self*)

**SHOW\_WARNING** = 0

**SHOW\_INFO** = 1

**SHOW\_ERROR** = 2

**SHOW\_DEBUG** = 3

**SHOW\_CURRENT**

**class InterfaceDevice**

Interface class that represents a device. Hidden feature: variables that need to be saved must be type-hinted: e.g.: *x*: int. See *obj\_to\_json()* for more info

**assign** (*self*, *machine\_to\_assign*, *resetAttribute=False*)  
Copy the attribute values of *machine\_to\_assign* to self. The references are not lost.

**Parameters**

- **machine\_to\_assign** – InterfaceDevice
- **resetAttribute** –

**class HowToPlotGraph** (*attribute\_x*, *attribute\_y*, *kwargs\_graph=None*, *excluded=None*)

**exclude\_col** (*self*, *id\_col*)  
Add *id\_col* to exclude from the graph

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

**class CollectionInfo** (*theCollection*, *kwargs*, *theID*)

**get\_collection** (*self*)

**get\_kwargs** (*self*)

**get\_id** (*self*)

**class LinkDataGraph**

**class \_collection\_linker**

**add\_link** (*self*, *idSlave*, *idMaster*)

```
    get_collection_master (self, idToGet)
    is_slave (self, idToCheck)
    set_same_master (self, idExistingSlave, idOtherSlave)
        Parameters
        • idExistingSlave – id collection of the existing slave
        • idOtherSlave – id collection of the new slave that has to be linked to an existing
          master
    add_collection (self, theCollection, kwargs=None)
    add_graph (self, howToPlotGraph)
    createGraphs (self)
    get_howToPlotGraph (self, idGraph)
    get_collectionInfo (self, idCollectionInfo)
    create_trace (self, collectionInfo, howToPlotGraph, idGraph)
    get_all_id_graphs (self)
    get_all_traces_id_graph (self, idGraph)
    update_graphs (self)
    is_slave (self, idGraph, idTrace)
    get_idCollection_from_graph (self, idGraph, idTrace, getMaster=True)
        From indices in the graph, get index of corresponding collection
    get_collection_from_graph (self, idGraph, idTrace, getMaster=True)
        From indices in the graph, get corresponding collection
    get_dataObject_from_graph (self, idGraph, idTrace, idPoint)
    get_dataObjects_from_graph (self, idGraph, idTrace, idPoint_list)
    remove_element_from_graph (self, idGraph, idTrace, idPoint, deleteFromMaster=False)
        Remove element from the graph, or the master collection
    remove_elements_from_trace (self, idGraph, idTrace, idPoints, deleteFromMaster=False)
        Performances      optimisation      when      compared      to      LinkDataGraph.remove\_element\_from\_graph\(\)
    link_collection_to_graph_collection (self, id_collection_graph, id_collection_master)
        Link data :param id_collection_graph: :param id_collection_master: :return:
    remove_trace (self, idGraph, idTrace)
    get_graph_and_trace_from_collection (self, idCollection)
        Reverse search: from a collection, get the associated graph
    get_mappingData_graph (self, idGraph)
    get_mappingData_trace (self, idGraph, idTrace)
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

class Options

    get_name (self, idOption)
    get_value (self, idOption)
    add_option (self, idOption, name, value)
    set_option (self, idOption, value)
    copy (self)
    set_self (self, the_options)
    __str__ (self)

class Option_class

    get_optionValue (self, optionId)
    set_optionValue (self, optionId, value)
    get_all_options (self)
    set_all_options (self, options)
    add_option (self, idOption, name, value)

```

## optimize

### Subpackages

### characterization

### characterization

## Module Contents

### class Characterization

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

`compute (self, theDevice)`

## interfaceCharacterization

### Module Contents

```
class InterfaceCharacterization
    Bases: optimeed.core.options.Option_class
    Interface for the evaluation of a device
    __str__(self)
```

### Package Contents

```
class InterfaceCharacterization
    Bases: optimeed.core.options.Option_class
    Interface for the evaluation of a device
    __str__(self)

class Characterization
    Bases: optimeed.optimize.characterization.interfaceCharacterization.
    InterfaceCharacterization
    compute(self, theDevice)
```

## mathsToPhysics

### interfaceMathsToPhysics

### Module Contents

```
class InterfaceMathsToPhysics
    Bases: optimeed.core.options.Option_class
    Interface to transform output from the optimizer to meaningful variables of the device
```

## mathsToPhysics

### Module Contents

```
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(self, xVector, theDevice, theOptimizationVariables)
    fromPhysToMaths(self, theDevice, theOptimizationVariables)
    __str__(self)
```



## Package Contents

### 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** (*self*, *xVector*, *theDevice*, *theOptimizationVariables*)

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

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

### class InterfaceMathsToPhysics

Bases: `optimeed.core.options.Option_class`

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

## objAndCons

### fastObjCons

## Module Contents

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

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

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

**compute** (*self*, *theDevice*)

**get\_name** (*self*)

### interfaceObjCons

## Module Contents

### class InterfaceObjCons

Bases: `optimeed.core.options.Option_class`

Interface class for objectives and constraints. The objective is to MINIMIZE and the constraint has to respect VALUE <= 0

**get\_name** (*self*)

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

## Package Contents

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

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

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

**compute** (*self*, *theDevice*)

```
get_name (self)
```

```
class InterfaceObjCons
```

```
    Bases: optimeed.core.options.Option_class
```

Interface class for objectives and constraints. The objective is to MINIMIZE and the constraint has to respect VALUE <= 0

```
get_name (self)
```

```
__str__ (self)
```

## optiAlgorithms

### Subpackages

#### convergence

#### evolutionaryConvergence

### Module Contents

```
class EvolutionaryConvergence (is_monobj=False)
```

```
    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]]]
```

```
is_monobj :bool
```

```
set_points_at_step (self, theStep, theObjectives_list, theConstraints_list)
```

```
get_pareto_convergence (self)
```

```
get_last_pareto (self)
```

```
get_hypervolume_convergence (self, refPoint=None)
```

Get the hypervolume indicator on each step

**Parameters** **refPoint** – Reference point needed to compute the hypervolume. If None is specified, uses the nadir point Example: [10, 10] for two objectives.

**Returns**

```
get_nb_objectives (self)
```

```
get_nadir_point (self)
```

```
get_nadir_point_all_steps (self)
```

```
get_nb_steps (self)
```

```
get_population_size (self)
```

```
get_graphs (self)
```

## hypervolume

## Module Contents

`__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** (*self*, *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** (*self*, *dimIndex*, *length*, *bounds*)

Recursive call to hypervolume calculation.

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

**preProcess** (*self*, *front*)

Sets up the list data structure needed for calculation.

**sortByDimension** (*self*, *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\_\_** (*self*)

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

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

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

**getLength** (*self*, *i*)

Returns the length of the i-th list.

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

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

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

Extends the list at the given index with the nodes.

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

Removes and returns 'node' from all lists in [0, 'index'].

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

Inserts 'node' at the position it had in all lists in [0, '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

#### class InterfaceConvergence

Simple interface to visually get the convergence of any optimization problem

### Package Contents

#### class EvolutionaryConvergence (*is\_monobj=False*)

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]]]

**is\_monobj** :bool

**set\_points\_at\_step** (*self, theStep, theObjectives\_list, theConstraints\_list*)

**get\_pareto\_convergence** (*self*)

**get\_last\_pareto** (*self*)

**get\_hypervolume\_convergence** (*self, refPoint=None*)

Get the hypervolume indicator on each step

**Parameters** **refPoint** – Reference point needed to compute the hypervolume. If None is specified, uses the nadir point Example: [10, 10] for two objectives.

#### Returns

**get\_nb\_objectives** (*self*)

**get\_nadir\_point** (*self*)

**get\_nadir\_point\_all\_steps** (*self*)

**get\_nb\_steps** (*self*)

**get\_population\_size** (*self*)

**get\_graphs** (*self*)

#### class InterfaceConvergence

Simple interface to visually get the convergence of any optimization problem

## NLOpt\_Algorithm

### Module Contents

#### class ConvergenceManager

**add\_point** (*self, newObj*)

**set\_pop\_size** (*self, popSize*)

```

class NLOpt_Algorithm
    Bases: optimeed.optimize.optiAlgorithms.algorithmInterface.
           AlgorithmInterface

    ALGORITHM = 0
    POPULATION_SIZE = 1

    compute (self, initialVectorGuess, listOfOptimizationVariables)

    set_evaluationFunction (self, evaluationFunction, callback_on_evaluate, numberOfObjectives,
                           _numberOfConstraints)

    set_maxtime (self, maxTime)

    __str__ (self)

    get_convergence (self)

```

algorithmInterface

## Module Contents

```

class AlgorithmInterface
    Bases: optimeed.core.options.Option_class

    Interface for the optimization algorithm

    reset (self)

```

multiObjective\_GA

## Module Contents

```

class MyConvergence (*args, **kwargs)
    Bases: optimeed.optimize.optiAlgorithms.convergence.InterfaceConvergence,
           optimeed.optimize.optiAlgorithms.platypus.core.Archive

    conv :EvolutionaryConvergence

    extend (self, solutions)

    get_graphs (self)

class MyProblem (theOptimizationVariables, nbr_objectives, nbr_constraints, evaluationFunction)
    Bases: optimeed.optimize.optiAlgorithms.platypus.core.Problem

    Automatically sets the optimization problem

    evaluate (self, solution)

class MyGenerator (initialVectorGuess)
    Bases: optimeed.optimize.optiAlgorithms.platypus.Generator

    Population generator to insert initial individual

    generate (self, problem)

class MyTerminationCondition (maxTime)
    Bases: optimeed.optimize.optiAlgorithms.platypus.core.TerminationCondition

```

```
    initialize (self, algorithm)
    shouldTerminate (self, algorithm)
class MyMapEvaluator (callback_on_evaluation)
    Bases: optimeed.optimize.optiAlgorithms.platypus.evaluator.Evaluator
    evaluate_all (self, jobs, **kwargs)
class MyMultiprocessEvaluator (callback_on_evaluation, numberOfCores)
    Bases: optimeed.optimize.optiAlgorithms.platypus.evaluator.Evaluator
    evaluate_all (self, jobs, **kwargs)
    close (self)
class MultiObjective_GA
    Bases: optimeed.optimize.optiAlgorithms.algorithmInterface.
AlgorithmInterface

    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

    compute (self, initialVectorGuess, listOfOptimizationVariables)
    set_evaluationFunction (self, evaluationFunction, callback_on_evaluation, numberOfObjec-
        tives, numberOfConstraints)

    set_maxtime (self, maxTime)
    __str__ (self)
    get_convergence (self)
```

## Package Contents

```
class MultiObjective_GA
    Bases: optimeed.optimize.optiAlgorithms.algorithmInterface.
AlgorithmInterface

    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

    compute (self, initialVectorGuess, listOfOptimizationVariables)
    set_evaluationFunction (self, evaluationFunction, callback_on_evaluation, numberOfObjec-
        tives, numberOfConstraints)

    set_maxtime (self, maxTime)
```

```

__str__(self)
get_convergence(self)

```

## optiVariable

### Module Contents

```

class OptimizationVariable (attributeName)
    Contains information about the optimization of a variable

    get_attribute_name (self)
        Return the attribute to set

    get_PhysToMaths (self, 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 (self, variableValue, deviceIn)
        Apply the value to the device

    __str__(self)

class Real_OptimizationVariable (attributeName, val_min, val_max)
    Bases: optimeed.optimize.optiVariable.OptimizationVariable
    Real (continuous) optimization variable. Most used type

    get_min_value (self)

    get_max_value (self)

    get_PhysToMaths (self, deviceIn)

    do_MathsToPhys (self, value, deviceIn)

    __str__(self)

class Binary_OptimizationVariable
    Bases: optimeed.optimize.optiVariable.OptimizationVariable
    Boolean (True/False) optimization variable.

    get_PhysToMaths (self, deviceIn)

    do_MathsToPhys (self, value, deviceIn)

    __str__(self)

class Integer_OptimizationVariable (attributeName, val_min, val_max)
    Bases: optimeed.optimize.optiVariable.OptimizationVariable
    Integer variable, in [min_value, max_value]

    get_min_value (self)

    get_max_value (self)

    get_PhysToMaths (self, deviceIn)

    do_MathsToPhys (self, value, deviceIn)

    __str__(self)

```

optimizer

## Module Contents

default

**class** PipeOptimization

Provides a live interface of the current optimization

**get\_device** (*self*)

Returns *InterfaceDevice* (not process safe, deprecated)

**get\_historic** (*self*)

Returns *OptiHistoric*

**set\_device** (*self*, *theDevice*)

**set\_historic** (*self*, *theHistoric*)

**class** OptiHistoric (*\*\*kwargs*)

Bases: object

Contains all the points that have been evaluated

**class** \_pointData (*currTime*, *objectives*, *constraints*)

**time** :float

**objectives** :List[float]

**constraints** :List[float]

**\_DEVICE** = autosaved

**\_LOGOPTI** = logopti

**\_RESULTS** = results

**\_CONVERGENCE** = optiConvergence

**add\_point** (*self*, *device*, *currTime*, *objectives*, *constraints*)

**set\_results** (*self*, *devicesList*)

**set\_convergence** (*self*, *theConvergence*)

**set\_info** (*self*, *theInfo*)

**save** (*self*)

**get\_results** (*self*)

**get\_convergence** (*self*)

Returns convergence *InterfaceConvergence*

**get\_devices** (*self*)

Returns List of devices (ordered by evaluation number)

**get\_logopti** (*self*)

Returns Log optimization (to check the convergence)



```

class Optimizer
    Bases: optimeed.core.options.Option_class

    Main optimizing class

    DISPLAY_INFO = 1

    KWARGS_OPTIHISTO = 2

    set_optimizer(self, theDevice, theObjectiveList, theConstraintList, theOptimization-
                Variables, theOptimizationAlgorithm=default['Algo'], theCharacteriza-
                tion=default['Charac'], theMathsToPhysics=default['M2P'])
        Prepare the optimizer for the optimization.

        Parameters

        • theDevice – object of type InterfaceDevice

        • theCharacterization – object of type InterfaceCharacterization

        • theMathsToPhysics – object of type InterfaceMathsToPhysics

        • theObjectiveList – list of objects of type InterfaceObjCons

        • theConstraintList – list of objects of type InterfaceObjCons

        • theOptimizationAlgorithm – list of objects of type AlgorithmInterface

        • theOptimizationVariables – list of objects of type OptimizationVariable

        Returns PipeOptimization

    run_optimization(self)
        Perform the optimization.

        Returns Collection of the best optimized machines

    set_max_opti_time(self, max_time_sec)

    evaluateObjectiveAndConstraints(self, x)
        Evaluates the performances of machine 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 actually 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. It is not (yet?) possible to access this motor on the main process before the end of the fork.
        This behaviour could be changed by using pipes or Managers.

        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)

    callback_on_evaluation(self, returnedValues)
        Save the output of evaluateObjectiveAndConstraints to optiHistoric. This function should be called by the
        optimizer IN a process safe context.

    formatInfo(self)

```

## Package Contents

```

class InterfaceCharacterization
    Bases: optimeed.core.options.Option_class

    Interface for the evaluation of a device

```

```
__str__(self)
```

**class Characterization**  
Bases: `optimeed.optimize.characterization.interfaceCharacterization.InterfaceCharacterization`  
`compute` (*self*, *theDevice*)

**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` (*self*, *xVector*, *theDevice*, *theOptimizationVariables*)  
`fromPhysToMaths` (*self*, *theDevice*, *theOptimizationVariables*)  
`__str__` (*self*)

**class InterfaceMathsToPhysics**  
Bases: `optimeed.core.options.Option_class`  
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` (*self*, *theDevice*)  
`get_name` (*self*)

**class InterfaceObjCons**  
Bases: `optimeed.core.options.Option_class`  
Interface class for objectives and constraints. The objective is to MINIMIZE and the constraint has to respect VALUE <= 0  
`get_name` (*self*)  
`__str__` (*self*)

**class MultiObjective\_GA**  
Bases: `optimeed.optimize.optiAlgorithms.algorithmInterface.AlgorithmInterface`  
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  
`compute` (*self*, *initialVectorGuess*, *listOfOptimizationVariables*)  
`set_evaluationFunction` (*self*, *evaluationFunction*, *callback\_on\_evaluation*, *numberOfObjectives*, *numberOfConstraints*)  
`set_maxtime` (*self*, *maxTime*)

```

    __str__(self)

    get_convergence(self)

class Real_OptimizationVariable(attributeName, val_min, val_max)
    Bases: optimeed.optimize.optiVariable.OptimizationVariable
    Real (continuous) optimization variable. Most used type

    get_min_value(self)

    get_max_value(self)

    get_PhysToMaths(self, deviceIn)

    do_MathsToPhys(self, value, deviceIn)

    __str__(self)

class Binary_OptimizationVariable
    Bases: optimeed.optimize.optiVariable.OptimizationVariable
    Boolean (True/False) optimization variable.

    get_PhysToMaths(self, deviceIn)

    do_MathsToPhys(self, value, deviceIn)

    __str__(self)

class Integer_OptimizationVariable(attributeName, val_min, val_max)
    Bases: optimeed.optimize.optiVariable.OptimizationVariable
    Integer variable, in [min_value, max_value]

    get_min_value(self)

    get_max_value(self)

    get_PhysToMaths(self, deviceIn)

    do_MathsToPhys(self, value, deviceIn)

    __str__(self)

class Optimizer
    Bases: optimeed.core.options.Option_class
    Main optimizing class

    DISPLAY_INFO = 1

    KWARGS_OPTIHISTO = 2

    set_optimizer(self, theDevice, theObjectiveList, theConstraintList, theOptimization-
        Variables, theOptimizationAlgorithm=default['Algo'], theCharacteriza-
        tion=default['Charac'], theMathsToPhysics=default['M2P'])
        Prepare the optimizer for the optimization.

    Parameters
        • theDevice – object of type InterfaceDevice
        • theCharacterization – object of type InterfaceCharacterization
        • theMathsToPhysics – object of type InterfaceMathsToPhysics
        • theObjectiveList – list of objects of type InterfaceObjCons
        • theConstraintList – list of objects of type InterfaceObjCons

```

- **theOptimizationAlgorithm** – list of objects of type *AlgorithmInterface*
- **theOptimizationVariables** – list of objects of type *OptimizationVariable*

**Returns** PipeOptimization

**run\_optimization** (*self*)

Perform the optimization.

**Returns** Collection of the best optimized machines

**set\_max\_opti\_time** (*self*, *max\_time\_sec*)

**evaluateObjectiveAndConstraints** (*self*, *x*)

Evaluates the performances of machine 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 actually 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. It is not (yet?) possible to access this motor on the main process before the end of the fork. This behaviour could be changed by using pipes or Managers.

**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)

**callback\_on\_evaluation** (*self*, *returnedValues*)

Save the output of *evaluateObjectiveAndConstraints* to *optiHistoric*. This function should be called by the optimizer IN a process safe context.

**formatInfo** (*self*)

**visualize**

**Subpackages**

**gui**

**Subpackages**

**widgets**

**Subpackages**

**graphsVisualWidget**

**Subpackages**

**examplesActionOnClick**

**on\_click\_anim**

## Module Contents

```

class DataAnimationOpenGL (theOpenGLWidget, theId=0, window_title='Animation')
    Bases: optimeed.visualize.gui.gui_data_animation.DataAnimationVisuals
    Implements DataAnimationVisuals to show opengl drawing
    update_widget_w_animation (self, key, index, the_data_animation)
    export_widget (self, painter)
    delete_key_widgets (self, key)

class DataAnimationOpenGLwText (*args, is_light=True, **kwargs)
    Bases: optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick.on_click_anim.DataAnimationOpenGL
    Implements DataAnimationVisuals to show opengl drawing and text
    update_widget_w_animation (self, key, index, the_data_animation)
    get_interesting_elements (self, devices_list)

class DataAnimationLines (is_light=True, theId=0, window_title='Animation')
    Bases: optimeed.visualize.gui.gui_data_animation.DataAnimationVisuals
    Implements DataAnimationVisuals to show drawing made out of lines (widget_line_drawer)
    export_widget (self, painter)
    delete_key_widgets (self, key)
    update_widget_w_animation (self, key, index, the_data_animation)
    get_interesting_elements (self, devices_list)

class DataAnimationVisualswText (is_light=True, theId=0, window_title='Animation')
    Bases: optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick.on_click_anim.DataAnimationLines
    Same as DataAnimationLines but also with text
    update_widget_w_animation (self, key, index, the_data_animation)

class on_graph_click_showAnim (theLinkDataGraph, theAnimation)
    Bases: optimeed.visualize.gui.widgets.widget_graphs_visual.on_graph_click_interface
    On click: add or remove an element to animate
    graph_clicked (self, theGraphVisual, index_graph, index_trace, indices_points)
    get_name (self)

```

**on\_click\_change\_symbol**

## Module Contents

```

class on_click_change_symbol (theLinkDataGraph)
    Bases: optimeed.visualize.gui.widgets.widget_graphs_visual.on_graph_click_interface
    On Click: Change the symbol of the point that is clicked

```

```
graph_clicked (self, theGraphVisual, index_graph, index_trace, indices_points)
get_name (self)
```

```
on_click_copy_something
```

## Module Contents

```
class on_click_copy_something (theDataLink, functionStrFromDevice)
    Bases: optimeed.visualize.gui.widgets.widget_graphs_visual.on_graph_click_interface
    On Click: copy something
    graph_clicked (self, the_graph_visual, index_graph, index_trace, indices_points)
    get_name (self)
```

```
on_click_delete
```

## Module Contents

```
class delete_gui
    Bases: PyQt5.QtWidgets.QMainWindow
class on_graph_click_delete (theDataLink)
    Bases: optimeed.visualize.gui.widgets.widget_graphs_visual.on_graph_click_interface
    On Click: Delete the points from the graph, and save the modified collection
    apply (self)
    reset (self)
    graph_clicked (self, theGraphVisual, index_graph, index_trace, indices_points)
    get_name (self)
```

```
on_click_export_collection
```

## Module Contents

```
class on_graph_click_export (theDataLink)
    Bases: optimeed.visualize.gui.widgets.widget_graphs_visual.on_graph_click_interface
    On click: export the selected points
    graph_clicked (self, theGraphVisual, index_graph, index_trace, indices_points)
    reset_graph (self)
    get_name (self)
```

`on_click_extract_pareto`

## Module Contents

```
class on_click_extract_pareto (theDataLink, max_x=False, max_y=False)
    Bases:                                     optimeed.visualize.gui.widgets.widget_graphs_visual.
                                                on_graph_click_interface
    On click: extract the pareto from the cloud of points
    graph_clicked (self, the_graph_visual, index_graph, index_trace, _)
    get_name (self)
```

`on_click_remove_trace`

## Module Contents

```
class on_graph_click_remove_trace (theDataLink)
    Bases:                                     optimeed.visualize.gui.widgets.widget_graphs_visual.
                                                on_graph_click_interface
    graph_clicked (self, theGraphVisual, index_graph, index_trace, _)
    get_name (self)
```

`on_click_showinfo`

## Module Contents

```
class on_graph_click_showInfo (theLinkDataGraph, visuals=None)
    Bases:                                     optimeed.visualize.gui.widgets.widget_graphs_visual.
                                                on_graph_click_interface
    On click: show informations about the points (loop through attributes)
    class DataInformationVisuals
        delete_visual (self, theVisual)
        add_visual (self, theVisual, theTrace, indexPoint)
        get_new_index (self)
        curr_index (self)
        graph_clicked (self, 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 (self)
class Repr_lines (attribute_lines)
    get_widget (self, theNewDevice)
```

```
class Repr_opengl (DeviceDrawer)
```

```
    get_widget (self, theNewDevice)
```

## Package Contents

```
class on_graph_click_delete (theDataLink)
```

```
    Bases:                                     optimeed.visualize.gui.widgets.widget_graphs_visual.  
                                                on_graph_click_interface
```

On Click: Delete the points from the graph, and save the modified collection

```
    apply (self)
```

```
    reset (self)
```

```
    graph_clicked (self, theGraphVisual, index_graph, index_trace, indices_points)
```

```
    get_name (self)
```

```
class on_graph_click_export (theDataLink)
```

```
    Bases:                                     optimeed.visualize.gui.widgets.widget_graphs_visual.  
                                                on_graph_click_interface
```

On click: export the selected points

```
    graph_clicked (self, theGraphVisual, index_graph, index_trace, indices_points)
```

```
    reset_graph (self)
```

```
    get_name (self)
```

```
class on_click_extract_pareto (theDataLink, max_x=False, max_y=False)
```

```
    Bases:                                     optimeed.visualize.gui.widgets.widget_graphs_visual.  
                                                on_graph_click_interface
```

On click: extract the pareto from the cloud of points

```
    graph_clicked (self, the_graph_visual, index_graph, index_trace, _)
```

```
    get_name (self)
```

```
class on_graph_click_showInfo (theLinkDataGraph, visuals=None)
```

```
    Bases:                                     optimeed.visualize.gui.widgets.widget_graphs_visual.  
                                                on_graph_click_interface
```

On click: show informations about the points (loop through attributes)

```
class DataInformationVisuals
```

```
    delete_visual (self, theVisual)
```

```
    add_visual (self, theVisual, theTrace, indexPoint)
```

```
    get_new_index (self)
```

```
    curr_index (self)
```

```
    graph_clicked (self, 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 (self)
```



```

class Repr_opengl (DeviceDrawer)

    get_widget (self, theNewDevice)

class Repr_lines (attribute_lines)

    get_widget (self, theNewDevice)

class on_graph_click_remove_trace (theDataLink)
    Bases:                                     optimeed.visualize.gui.widgets.widget_graphs_visual.
                                             on_graph_click_interface

    graph_clicked (self, theGraphVisual, index_graph, index_trace, _)

    get_name (self)

class on_click_copy_something (theDataLink, functionStrFromDevice)
    Bases:                                     optimeed.visualize.gui.widgets.widget_graphs_visual.
                                             on_graph_click_interface

    On Click: copy something

    graph_clicked (self, the_graph_visual, index_graph, index_trace, indices_points)

    get_name (self)

class on_click_change_symbol (theLinkDataGraph)
    Bases:                                     optimeed.visualize.gui.widgets.widget_graphs_visual.
                                             on_graph_click_interface

    On Click: Change the symbol of the point that is clicked

    graph_clicked (self, theGraphVisual, index_graph, index_trace, indices_points)

    get_name (self)

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

class DataAnimationVisuals (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 (self, 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 (self, trace_id, indexPoint)

    delete_point (self, trace_id, thePoint)

```

```
    reset_all (self)
    delete_all (self)
    pause_play (self)
    show_all (self)
    next_frame (self)
    slider_handler (self)
    frame_selector (self)
    set_refreshTime (self)
    is_empty (self)
    run (self)
    closeEvent (self, _)
    contains_trace (self, trace_id)
    export_picture (self)

class widget_text (theText, is_light=False, convertToHtml=False)
    Bases: PyQt5.QtWidgets.QLabel
    Widget able to display a text

    set_text (self, theText, convertToHtml=False)
        Set the text to display

class widget_line_drawer (minWinHeight=300, minWinWidth=300, is_light=True)
    Bases: PyQt5.QtWidgets.QWidget
    Widget allowing to display several lines easily

    signal_must_update
    on_update_signal (self, listOfLines)
    delete_lines (self, key_id)
        Delete the lines :param key_id: id to delete :return:
    set_lines (self, listOfLines, key_id=0, pen=None)
        Set the lines to display :param listOfLines: list of [x1, y1, z1, x2, y2, z2] corresponding to lines :param
        key_id: id of the trace :param pen: pen used to draw the lines :return:
    paintEvent (self, event, painter=None)
    get_extrema_lines (self)

class DataAnimationOpenGL (theOpenGLWidget, theId=0, window_title='Animation')
    Bases: optimeed.visualize.gui.gui_data_animation.DataAnimationVisuals
    Implements DataAnimationVisuals to show opengl drawing

    update_widget_w_animation (self, key, index, the_data_animation)
    export_widget (self, painter)
    delete_key_widgets (self, key)

class DataAnimationOpenGLwText (*args, is_light=True, **kwargs)
    Bases:
        optimeed.visualize.gui.widgets.graphsVisualWidget.
        examplesActionOnClick.on_click_anim.DataAnimationOpenGL
```

Implements *DataAnimationVisuals* to show opengl drawing and text

**update\_widget\_w\_animation** (*self, key, index, the\_data\_animation*)

**get\_interesting\_elements** (*self, devices\_list*)

**class DataAnimationLines** (*is\_light=True, theId=0, window\_title='Animation'*)

Bases: *optimeed.visualize.gui.gui\_data\_animation.DataAnimationVisuals*

Implements *DataAnimationVisuals* to show drawing made out of lines (*widget\_line\_drawer*)

**export\_widget** (*self, painter*)

**delete\_key\_widgets** (*self, key*)

**update\_widget\_w\_animation** (*self, key, index, the\_data\_animation*)

**get\_interesting\_elements** (*self, devices\_list*)

**class DataAnimationVisualswText** (*is\_light=True, theId=0, window\_title='Animation'*)

Bases: *optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick.on\_click\_anim.DataAnimationLines*

Same as *DataAnimationLines* but also with text

**update\_widget\_w\_animation** (*self, key, index, the\_data\_animation*)

**class on\_graph\_click\_showAnim** (*theLinkDataGraph, theAnimation*)

Bases: *optimeed.visualize.gui.widgets.widget\_graphs\_visual.on\_graph\_click\_interface*

On click: add or remove an element to animate

**graph\_clicked** (*self, theGraphVisual, index\_graph, index\_trace, indices\_points*)

**get\_name** (*self*)

## graphVisual

## Module Contents

**class GraphVisual** (*theWidgetGraphVisual*)

Provide an interface to a graph. A graph contains traces.

**set\_fontTicks** (*self, fontSize, fontname=None*)

Set font of the ticks

### Parameters

- **fontSize** – Size of the font
- **fontname** – Name of the font

**set\_numberTicks** (*self, 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** (*self*, *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** (*self*)  
Get the legend

**get\_axis** (*self*, *axis*)  
Get the axis

**Parameters** **axis** – Axis (string, “bottom”, “left”, “right”, “top”)

**Returns** axis object

**set\_fontLegend** (*self*, *font\_size*, *font\_color*, *fontname*=None)

**set\_label\_pos** (*self*, *orientation*, *x\_offset*=0, *y\_offset*=0)

**set\_color\_palette** (*self*, *palette*)

**apply\_palette** (*self*)

**hide\_axes** (*self*)

**add\_feature** (*self*, *theFeature*)  
To add any pyqtgraph item to the graph

**remove\_feature** (*self*, *theFeature*)  
To remove any pyqtgraph item from the graph

**add\_data** (*self*, *idGraph*, *theColor*, *theData*)

**set\_graph\_properties** (*self*, *theTrace*)  
This function is automatically called on creation of the graph

**set\_lims** (*self*, *xlim*, *ylim*)  
Set limits of the graphs, xlim or ylim = [val\_low, val\_high]. Or None.

**add\_trace** (*self*, *idTrace*, *theTrace*)  
Add a TraceVisual to the graph, with index idTrace

**set\_legend** (*self*)  
Set default legend options (color and font)

**set\_title** (*self*, *titleName*, *\*\*kwargs*)  
Set title of the graph

**Parameters** **titleName** – title to set

**get\_trace** (*self*, *idTrace*)  
Return the TraceVisual correspondong to the index idTrace

**get\_all\_traces** (*self*)  
Return a dictionary {idtrace: TraceVisual}.

**delete\_trace** (*self*, *idTrace*)  
Delete the trace of index idTrace

**delete** (*self*)  
Delete the graph

**linkXToGraph** (*self*, *graph*)

Link the axis of the current graph to an other *GraphVisual*

**update** (*self*)

Update the traces contained in the graph

**fast\_update** (*self*)

Same as *update* () but faster. This is NOT thread safe (cannot be called a second time before finishing operation)

**axis\_equal** (*self*)

**grid\_off** (*self*)

Turn off grid

## pyqtgraphRedefine

### Module Contents

#### isOnWindows

Other modified files (directly): ScatterPlotItem.py, to change point selection. Ctrl + clic: select area. Clic: only one single point

**class myGraphicsLayoutWidget** (*parent=None*, *\*\*\_kwargs*)

Bases: *optimeed.visualize.gui.widgets.graphsVisualWidget.pyqtgraph.GraphicsView*

**useOpenGL** (*self*, *b=True*)

Overwrited to fix bad antialiasing while using openGL

**class myGraphicsLayout**

Bases: *optimeed.visualize.gui.widgets.graphsVisualWidget.pyqtgraph.GraphicsLayout*

**addItem** (*self*, *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** (*self*, *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.gui.widgets.graphsVisualWidget.pyqtgraph.graphicsItems.LegendItem.ItemSample*

**set\_offset** (*self*, *offset*)

**set\_width\_cell** (*self*, *width*)

**paint** (*self*, *p*, *\*args*)

Overwrites to make matlab-like samples

**class myLegend** (*size=None*, *offset=(30, 30)*, *is\_light=False*)

Bases: `optimeed.visualize.gui.widgets.graphsVisualWidget.pyqtgraph.LegendItem`

Legend that fixes bugs (flush left + space) from pyqtgraph's legend

**set\_space\_sample\_label** (*self*, *theSpace*)

To set the gap between the sample and the label

**set\_offset\_sample** (*self*, *offset*)

To tune the offset between the sample and the text

**set\_width\_cell\_sample** (*self*, *width*)

Set width of sample

**updateSize** (*self*)

**addItem** (*self*, *item*, *name*)

Overwrites to flush left

**apply\_width\_sample** (*self*)

**set\_font** (*self*, *font\_size*, *font\_color*, *fontname=None*)

**paint** (*self*, *p*, *\*args*)

Overwrited to select background color

**set\_position** (*self*, *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**

Bases: `optimeed.visualize.gui.widgets.graphsVisualWidget.pyqtgraph.LabelItem`

**setText** (*self*, *text*, *\*\*args*)

Overwrited to add font-family to options

**class myAxis** (*orientation*)

Bases: `optimeed.visualize.gui.widgets.graphsVisualWidget.pyqtgraph.AxisItem`

**get\_label\_pos** (*self*)

Overwrited to place label closer to the axis

**resizeEvent** (*self*, *ev=None*)

Overwrited to place label closer to the axis

**set\_label\_pos** (*self*, *orientation*, *x\_offset=0*, *y\_offset=0*)

**set\_number\_ticks** (*self*, *number*)

`smallGui`

## Module Contents

**class** `guiPyqtgraph` (*graphsVisual*, *\*\*kwargs*)

Create a gui for pyqtgraph with trace selection options, export and action on clic choices

**refreshTraceList** (*self*)

Refresh all the traces

`traceVisual`

## Module Contents

**class** `TraceVisual` (*theColor*, *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** (*self*, *index*, *newPaintElem*)

**modify\_paintElems** (*self*, *paintElemsIn\_List*)

Apply transformation to paintElemsIn\_List

**Parameters** `paintElemsIn_List` – list of brushes or pens to modify

**Returns** False if nothing has been modified, True is something has been modified

**reset\_paintElem** (*self*, *index*)

Remove transformation of point index

**reset** (*self*)

**signal\_must\_update**

**hide\_points** (*self*)

Hide all the points

**get\_color** (*self*)

Get colour of the trace, return tuple (r,g,b)

**set\_color** (*self*, *color*)

Set colour of the trace, argument as tuple (r,g,b)

**get\_base\_symbol\_brush** (*self*)

Get symbol brush configured for this trace, return `pg.QBrush`

**get\_base\_pen** (*self*)

Get pen configured for this trace, return `pg.QPen`

**get\_base\_symbol\_pen** (*self*)

Get symbol pen configured for this trace, return `pg.QPen`

**get\_base\_symbol** (*self*)

Get base symbol configured for this trace, return str of the symbol (e.g. 'o')

**get\_symbol** (*self*, *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** (*self*)

Forces the trace to refresh.

**get\_length** (*self*)

Return number of data to plot

**hide** (*self*)

Hides the trace

**show** (*self*)

Shows the trace

**toggle** (*self*, *boolean*)

Toggle the trace (hide/show)

**get\_data** (*self*)

Get data to plot `Data`

**get\_brushes** (*self*, *size*)

Get actual brushes for the trace (=symbol filling). return a list which maps each points to a symbol brush

**set\_brush** (*self*, *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** (*self*, *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** (*self*, *list\_indexPoint*, *list\_newbrush*)

Same as `set_brush()` but by taking a list as input

**reset\_brush** (*self*, *indexPoint*, *update=True*)

Reset the brush of the point `indexpoint`

**reset\_all\_brushes** (*self*)

Reset all the brushes

**reset\_symbol** (*self*, *indexPoint*, *update=True*)

Reset the symbol shape of the point `indexpoint`

**get\_symbolPens** (*self*, *size*)

Get actual symbol pens for the trace (=symbol outline). return a list which maps each points to a symbol pen

**set\_symbolPen** (*self*, *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** (*self*, *list\_indexPoint*, *list\_newpens*)  
Same as *set\_symbolPen* () but by taking a list as input

**reset\_symbolPen** (*self*, *indexPoint*)  
Reset the symbol pen of the point indexpoint

**reset\_all\_symbolPens** (*self*)  
Reset all the symbol pens

## OpenGLWidget

### ContextHandler

### Module Contents

**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** (*self*, *theSpecialButtonsMapping*)

**set\_deviceDrawer** (*self*, *theDeviceDrawer*)

**set\_deviceToDraw** (*self*, *theDeviceToDraw*)

**resizeWindowAction** (*self*, *new\_width*, *new\_height*)

**mouseWheelAction** (*self*, *deltaAngle*)

**mouseClicAction** (*self*, *button*, *my\_x*, *y*)

**mouseMotionAction** (*self*, *my\_x*, *y*)

**keyboardPushAction** (*self*, *key*)

**keyboardReleaseAction** (*self*, *key*, *my\_x*, *y*)

**\_\_draw\_axis\_\_** (*self*)

**redraw** (*self*)

**get\_text\_to\_write** (*self*)

**\_\_lightingInit\_\_** (*self*)

**initialize** (*self*)

```
__reset__(self)
```

## DeviceDrawerInterface

### Module Contents

```
class DeviceDrawerInterface
```

```
    keyboard_push_action(self, theKey)
```

```
    get_colour_scalebar(self)
```

```
    get_colour_background(self)
```

```
    get_opengl_options(self)
```

## Materials\_visual

### Module Contents

```
class MaterialRenderingProperties(amb3, dif3, spec3, shin)
```

```
    __spec3__ = [0, 0, 0, 0]
```

```
    __dif3__ = [0, 0, 0, 0]
```

```
    __amb3__ = [0, 0, 0, 0]
```

```
    __shin__ = 0
```

```
    getSpec3(self)
```

```
    getDif3(self)
```

```
    getAmb3(self)
```

```
    getShin(self)
```

```
    activateMaterialProperties(self, alpha=1)
```

```
Emerald_material
```

```
Yellow_Emerald_material
```

```
Brass_material
```

```
Bronze_material
```

```
Silver_material
```

```
Steel_material
```

```
Copper_material
```

```
Chrome_material
```

```
Blue_material
```

```
Red_material
```

## OpenGLFunctions\_Library

### Module Contents

**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*)

## TriangulatePolygon

### Module Contents

**IsConvex** (*a, b, c*)  
**InTriangle** (*a, b, c, p*)  
**IsClockwise** (*poly*)  
**GetEar** (*poly*)  
**reformatXYtoList** (*xList, yList*)  
**meshPolygon** (*xList, yList*)

## quaternions

### Module Contents

**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*)

## widget\_graphs\_visual

### Module Contents

**class on\_graph\_click\_interface**

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

**class widget\_graphs\_visual** (*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** (*self*, *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** (*self*, *idGraph*)

**\_\_check\_graphs** (*self*)

**on\_click** (*self*, *plotDataItem*, *clicked\_points*)

**update\_graphs** (*self*, *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** (*self*)

Use this method to update the graph in a fast way. NOT THREAD SAFE.

**exportGraphs** (*self*)

Export the graphs

**link\_axes** (*self*)

**get\_graph** (*self*, *idGraph*)

Get corresponding `GraphVisual` of the graph `idGraph`

**keyPressEvent** (*self*, *event*)

What happens if a key is pressed. R: reset the axes to their default value

**delete\_graph** (*self*, *idGraph*)

Delete the graph `idGraph`

**delete** (*self*)

**get\_all\_graphsVisual** (*self*)

Return a dictionary {`idGraph`: `GraphVisual`}.

**get\_layout\_buttons** (*self*)

Get the `QGraphicsLayout` where it's possible to add buttons, etc.

**set\_actionOnClick** (*self*, *theActionOnClick*)

Action to perform when the graph is clicked

**Parameters** `theActionOnClick` – *on\_graph\_click\_interface*

**Returns**

**set\_title** (*self*, *idGraph*, *titleName*, *\*\*kwargs*)

Set title of the graph

**Parameters**

- **idGraph** – id of the graph
- **titleName** – title to set

**set\_article\_template** (*self*, *graph\_size\_x=8.8*, *graph\_size\_y=4.4*, *legendPosition='NW'*)

Method to set the graphs to article quality graph.

**Parameters**

- **graph\_size\_x** – width of the graph in cm
- **graph\_size\_y** – height of the graph in cm
- **legendPosition** – position of the legend (NE, SE, SW, NW)

**Returns**

`widget_line_drawer`

## Module Contents

**class** `widget_line_drawer` (*minWinHeight=300*, *minWinWidth=300*, *is\_light=True*)

Bases: `PyQt5.QtWidgets.QWidget`

Widget allowing to display several lines easily

**signal\_must\_update**

**on\_update\_signal** (*self*, *listOfLines*)

**delete\_lines** (*self*, *key\_id*)

Dele the lines :param key\_id: id to delete :return:

**set\_lines** (*self*, *listOfLines*, *key\_id=0*, *pen=None*)

Set the lines to display :param listOfLines: list of [x1, y1, z1, x2, y2, z2] corresponding to lines :param key\_id: id of the trace :param pen: pen used to draw the lines :return:

**paintEvent** (*self*, *event*, *painter=None*)

**get\_extrema\_lines** (*self*)

**widget\_menuButton**

## Module Contents

**class widget\_menuButton** (*theParentButton*)

Bases: `PyQt5.QtWidgets.QMenu`

Same as `QMenu`, but integrates it behind a button more easily.

**showEvent** (*self*, *QShowEvent*)

**widget\_openGL**

## Module Contents

**class widget\_openGL** (*parent=None*)

Bases: `PyQt5.QtWidgets.QOpenGLWidget`

Interface that provides opengl capabilities. Ensures zoom, light, rotation, etc.

**sizeHint** (*self*)

**minimumSizeHint** (*self*)

**set\_deviceDrawer** (*self*, *theDeviceDrawer*)

Set a drawer `optimeed.visualize.gui.widgets.openGLWidget.DeviceDrawerInterface.DeviceDrawerInterface`

**set\_deviceToDraw** (*self*, *theDeviceToDraw*)

Set the device to draw `optimeed.InterfaceDevice.InterfaceDevice`

**initializeGL** (*self*)

**paintGL** (*self*)

**resizeGL** (*self*, *w*, *h*)

**mousePressEvent** (*self*, *event*)

**mouseMoveEvent** (*self*, *event*)

**keyPressEvent** (*self*, *event*)

**wheelEvent** (*self*, *QWheelEvent*)

## widget\_text

### Module Contents

**class widget\_text** (*theText, is\_light=False, convertToHtml=False*)

Bases: `PyQt5.QtWidgets.QLabel`

Widget able to display a text

**set\_text** (*self, theText, convertToHtml=False*)

Set the text to display

**class scrollable\_widget\_text** (*theText, is\_light=False, convertToHtml=False*)

Bases: `PyQt5.QtWidgets.QWidget`

Same as `widget_text` but scrollable

**set\_text** (*self, theText, convertToHtml=False*)

### Package Contents

**class widget\_graphs\_visual** (*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** (*self, 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** (*self, idGraph*)

**\_\_check\_graphs** (*self*)

**on\_click** (*self, plotDataItem, clicked\_points*)

**update\_graphs** (*self, 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** (*self*)

Use this method to update the graph in a fast way. NOT THREAD SAFE.

**exportGraphs** (*self*)

Export the graphs

**link\_axes** (*self*)

**get\_graph** (*self*, *idGraph*)

Get corresponding `GraphVisual` of the graph *idGraph*

**keyPressEvent** (*self*, *event*)

What happens if a key is pressed. R: reset the axes to their default value

**delete\_graph** (*self*, *idGraph*)

Delete the graph *idGraph*

**delete** (*self*)

**get\_all\_graphsVisual** (*self*)

Return a dictionary {*idGraph*: `GraphVisual`}.

**get\_layout\_buttons** (*self*)

Get the `QGraphicsLayout` where it's possible to add buttons, etc.

**set\_actionOnClick** (*self*, *theActionOnClick*)

Action to perform when the graph is clicked

**Parameters** **theActionOnClick** – *on\_graph\_click\_interface*

**Returns**

**set\_title** (*self*, *idGraph*, *titleName*, *\*\*kwargs*)

Set title of the graph

**Parameters**

- **idGraph** – id of the graph
- **titleName** – title to set

**set\_article\_template** (*self*, *graph\_size\_x*=8.8, *graph\_size\_y*=4.4, *legendPosition*='NW')

Method to set the graphs to article quality graph.

**Parameters**

- **graph\_size\_x** – width of the graph in cm
- **graph\_size\_y** – height of the graph in cm
- **legendPosition** – position of the legend (NE, SE, SW, NW)

**Returns**

**class widget\_line\_drawer** (*minWinHeight*=300, *minWinWidth*=300, *is\_light*=True)

Bases: `PyQt5.QtWidgets.QWidget`

Widget allowing to display several lines easily

**signal\_must\_update**

**on\_update\_signal** (*self*, *listOfLines*)

**delete\_lines** (*self*, *key\_id*)

Dele the lines :param *key\_id*: id to delete :return:

**set\_lines** (*self*, *listOfLines*, *key\_id*=0, *pen*=None)

Set the lines to display :param *listOfLines*: list of [*x1*, *y1*, *z1*, *x2*, *y2*, *z2*] corresponding to lines :param *key\_id*: id of the trace :param *pen*: pen used to draw the lines :return:



```

    paintEvent (self, event, painter=None)
    get_extrema_lines (self)
class widget_menuButton (theParentButton)
    Bases: PyQt5.QtWidgets.QMenu
    Same as QMenu, but integrates it behind a button more easily.
    showEvent (self, QShowEvent)
class widget_opengl (parent=None)
    Bases: PyQt5.QtWidgets.QOpenGLWidget
    Interface that provides opengl capabilities. Ensures zoom, light, rotation, etc.
    sizeHint (self)
    minimumSizeHint (self)
    set_deviceDrawer (self, theDeviceDrawer)
        Set a drawer optimeed.visualize.gui.widgets.openglWidget.DeviceDrawerInterface.DeviceDrawerInterface
    set_deviceToDraw (self, theDeviceToDraw)
        Set the device to draw optimeed.InterfaceDevice.InterfaceDevice
    initializeGL (self)
    paintGL (self)
    resizeGL (self, w, h)
    mousePressEvent (self, event)
    mouseMoveEvent (self, event)
    keyPressEvent (self, event)
    wheelEvent (self, QWheelEvent)
class widget_text (theText, is_light=False, convertToHtml=False)
    Bases: PyQt5.QtWidgets.QLabel
    Widget able to display a text
    set_text (self, theText, convertToHtml=False)
        Set the text to display
class on_graph_click_delete (theDataLink)
    Bases: optimeed.visualize.gui.widgets.widget_graphs_visual.on_graph_click_interface
    On Click: Delete the points from the graph, and save the modified collection
    apply (self)
    reset (self)
    graph_clicked (self, theGraphVisual, index_graph, index_trace, indices_points)
    get_name (self)
class on_graph_click_export (theDataLink)
    Bases: optimeed.visualize.gui.widgets.widget_graphs_visual.on_graph_click_interface
    On click: export the selected points

```

```
graph_clicked (self, theGraphVisual, index_graph, index_trace, indices_points)

reset_graph (self)

get_name (self)

class on_click_extract_pareto (theDataLink, max_x=False, max_y=False)
    Bases: optimeed.visualize.gui.widgets.widget_graphs_visual.
            on_graph_click_interface
    On click: extract the pareto from the cloud of points

    graph_clicked (self, the_graph_visual, index_graph, index_trace, _)

    get_name (self)

class on_graph_click_showInfo (theLinkDataGraph, visuals=None)
    Bases: optimeed.visualize.gui.widgets.widget_graphs_visual.
            on_graph_click_interface
    On click: show informations about the points (loop through attributes)

class DataInformationVisuals

    delete_visual (self, theVisual)

    add_visual (self, theVisual, theTrace, indexPoint)

    get_new_index (self)

    curr_index (self)

    graph_clicked (self, 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 (self)

class Repr_opengl (DeviceDrawer)

    get_widget (self, theNewDevice)

class Repr_lines (attribute_lines)

    get_widget (self, theNewDevice)

class on_graph_click_remove_trace (theDataLink)
    Bases: optimeed.visualize.gui.widgets.widget_graphs_visual.
            on_graph_click_interface

    graph_clicked (self, theGraphVisual, index_graph, index_trace, _)

    get_name (self)

class on_click_copy_something (theDataLink, functionStrFromDevice)
    Bases: optimeed.visualize.gui.widgets.widget_graphs_visual.
            on_graph_click_interface
    On Click: copy something

    graph_clicked (self, the_graph_visual, index_graph, index_trace, indices_points)

    get_name (self)
```

```

class on_click_change_symbol (theLinkDataGraph)
    Bases: optimeed.visualize.gui.widgets.widget_graphs_visual.
            on_graph_click_interface

    On Click: Change the symbol of the point that is clicked

    graph_clicked (self, theGraphVisual, index_graph, index_trace, indices_points)

    get_name (self)

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

class DataAnimationVisuals (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 (self, 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 (self, trace_id, indexPoint)

    delete_point (self, trace_id, thePoint)

    reset_all (self)

    delete_all (self)

    pause_play (self)

    show_all (self)

    next_frame (self)

    slider_handler (self)

    frame_selector (self)

    set_refreshTime (self)

    is_empty (self)

    run (self)

    closeEvent (self, _)

    contains_trace (self, trace_id)

    export_picture (self)

```

```
class widget_text (theText, is_light=False, convertToHtml=False)
    Bases: PyQt5.QtWidgets.QLabel

    Widget able to display a text

    set_text (self, theText, convertToHtml=False)
        Set the text to display

class widget_line_drawer (minWinHeight=300, minWinWidth=300, is_light=True)
    Bases: PyQt5.QtWidgets.QWidget

    Widget allowing to display several lines easily

    signal_must_update

    on_update_signal (self, listOfLines)

    delete_lines (self, key_id)
        Delete the lines :param key_id: id to delete :return:

    set_lines (self, listOfLines, key_id=0, pen=None)
        Set the lines to display :param listOfLines: list of [x1, y1, z1, x2, y2, z2] corresponding to lines :param
        key_id: id of the trace :param pen: pen used to draw the lines :return:

    paintEvent (self, event, painter=None)

    get_extrema_lines (self)

class DataAnimationOpenGL (theOpenGLWidget, theId=0, window_title='Animation')
    Bases: optimeed.visualize.gui.gui_data_animation.DataAnimationVisuals

    Implements DataAnimationVisuals to show opengl drawing

    update_widget_w_animation (self, key, index, the_data_animation)

    export_widget (self, painter)

    delete_key_widgets (self, key)

class DataAnimationOpenGLwText (*args, is_light=True, **kwargs)
    Bases: optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick.on_click_anim.DataAnimationOpenGL

    Implements DataAnimationVisuals to show opengl drawing and text

    update_widget_w_animation (self, key, index, the_data_animation)

    get_interesting_elements (self, devices_list)

class DataAnimationLines (is_light=True, theId=0, window_title='Animation')
    Bases: optimeed.visualize.gui.gui_data_animation.DataAnimationVisuals

    Implements DataAnimationVisuals to show drawing made out of lines (widget_line_drawer)

    export_widget (self, painter)

    delete_key_widgets (self, key)

    update_widget_w_animation (self, key, index, the_data_animation)

    get_interesting_elements (self, devices_list)

class DataAnimationVisualswText (is_light=True, theId=0, window_title='Animation')
    Bases: optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick.on_click_anim.DataAnimationLines

    Same as DataAnimationLines but also with text
```

```

    update_widget_w_animation (self, key, index, the_data_animation)
class on_graph_click_showAnim (theLinkDataGraph, theAnimation)
    Bases: optimeed.visualize.gui.widgets.widget_graphs_visual.
on_graph_click_interface
    On click: add or remove an element to animate
    graph_clicked (self, theGraphVisual, index_graph, index_trace, indices_points)
    get_name (self)
class guiPyqtgraph (graphsVisual, **kwargs)
    Create a gui for pyqtgraph with trace selection options, export and action on clic choices
    refreshTraceList (self)
        Refresh all the traces
class DeviceDrawerInterface

    keyboard_push_action (self, theKey)
    get_colour_scalebar (self)
    get_colour_background (self)
    get_opengl_options (self)

```

*gui\_collection\_exporter*

## Module Contents

```

class gui_collection_exporter
    Bases: PyQt5.QtWidgets.QMainWindow
    Simple gui that allows to export data
    signal_has_exported
    signal_has_reset
    exportCollection (self)
        Export the collection
    reset (self)
    add_data_to_collection (self, data)
        Add data to the collection to export
        Parameters data – Whichever type you like
    set_info (self, info)
    set_collection (self, theCollection)

```

*gui\_data\_animation*

## Module Contents

**class DataAnimationTrace** (*elements\_list, theTrace*)

Contains all the element to animate for a trace

**class element\_animation** (*elements*)

**get** (*self*)

**get\_element\_animations** (*self, 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** (*self*)

**delete\_all** (*self*)

**get\_indices\_to\_show** (*self*)

**add\_element** (*self, indexPoint*)

**add\_index\_to\_show** (*self, index*)

**\_remove\_index\_from\_show** (*self, index*)

**set\_curr\_brush** (*self, index\_in\_show*)

**set\_idle\_brush** (*self, index\_in\_show*)

**get\_number\_of\_elements** (*self*)

**map\_index** (*self, index\_in\_show*)

**get\_base\_pen** (*self*)

**class DataAnimationVisuals** (*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** (*self, 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** (*self, trace\_id, indexPoint*)

**delete\_point** (*self, trace\_id, thePoint*)

**reset\_all** (*self*)

**delete\_all** (*self*)

**pause\_play** (*self*)

```

show_all (self)
next_frame (self)
slider_handler (self)
frame_selector (self)
set_refreshTime (self)
is_empty (self)
run (self)
closeEvent (self, _)
contains_trace (self, trace_id)
export_picture (self)

```

```
gui_data_selector
```

## Module Contents

```
app
```

```
class Action_on_selector_update
```

```
class Attribute_selector (attribute_name, value)
```

```

add_child (self, child)
get_children (self)
get_name (self)
get_min_max_attributes (self)
__str__ (self)

```

```
class Container_attribute_selector (containerName)
```

```

add_child (self, child)
add_attribute_selector (self, attribute_selector)
set_attribute_selectors (self, attribute_selectors)
get_name (self)
get_children (self)
get_attribute_selectors (self)
__str__ (self)

```

```

class GuiDataSelector (collections_in: CollectionsToVisualise, actionOnUpdate: Action_on_selector_update)
Bases: PyQt5.QtWidgets.QMainWindow

```

```

theActionOnUpdate
    Generate GUI
apply_filters (self, _)

```

```
run (self)
is_object_selected (container_in, object_in)
check_and_add_if_float (the_container, attribute_value, attribute_name, parent=None)
manage_list (the_container, in_object, _listOfValues, _listName)
get_attr_object (the_container, in_object)

gui_mainWindow
```

## Module Contents

### 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

class gui_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 (self, actionOnWindowClosed)

    closeEvent (self, event)

    run (self, hold=False)
        Display the window

    keyPressEvent (self, event)
```

## Package Contents

```
class gui_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 (self, actionOnWindowClosed)

    closeEvent (self, event)

    run (self, hold=False)
        Display the window

    keyPressEvent (self, event)

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
```



```

class gui_collection_exporter
    Bases: PyQt5.QtWidgets.QMainWindow

    Simple gui that allows to export data

    signal_has_exported

    signal_has_reset

    exportCollection (self)
        Export the collection

    reset (self)

    add_data_to_collection (self, data)
        Add data to the collection to export

        Parameters data – Whichever type you like

    set_info (self, info)

    set_collection (self, theCollection)

class DataAnimationVisuals (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 (self, 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 (self, trace_id, indexPoint)

    delete_point (self, trace_id, thePoint)

    reset_all (self)

    delete_all (self)

    pause_play (self)

    show_all (self)

    next_frame (self)

    slider_handler (self)

    frame_selector (self)

    set_refreshTime (self)

    is_empty (self)

    run (self)

```

**closeEvent** (*self*, *\_*)

**contains\_trace** (*self*, *trace\_id*)

**export\_picture** (*self*)

**class widget\_graphs\_visual** (*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** (*self*, *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** (*self*, *idGraph*)

**\_\_check\_graphs** (*self*)

**on\_click** (*self*, *plotDataItem*, *clicked\_points*)

**update\_graphs** (*self*, *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** (*self*)

Use this method to update the graph in a fast way. NOT THREAD SAFE.

**exportGraphs** (*self*)

Export the graphs

**link\_axes** (*self*)

**get\_graph** (*self*, *idGraph*)

Get corresponding *GraphVisual* of the graph *idGraph*

**keyPressEvent** (*self*, *event*)

What happens if a key is pressed. R: reset the axes to their default value

**delete\_graph** (*self*, *idGraph*)

Delete the graph *idGraph*

**delete** (*self*)

**get\_all\_graphsVisual** (*self*)

Return a dictionary {*idGraph*: *GraphVisual*}.

**get\_layout\_buttons** (*self*)

Get the QGraphicsLayout where it's possible to add buttons, etc.

**set\_actionOnClick** (*self, theActionOnClick*)

Action to perform when the graph is clicked

**Parameters** **theActionOnClick** – *on\_graph\_click\_interface*

**Returns**

**set\_title** (*self, idGraph, titleName, \*\*kwargs*)

Set title of the graph

**Parameters**

- **idGraph** – id of the graph
- **titleName** – title to set

**set\_article\_template** (*self, graph\_size\_x=8.8, graph\_size\_y=4.4, legendPosition='NW'*)

Method to set the graphs to article quality graph.

**Parameters**

- **graph\_size\_x** – width of the graph in cm
- **graph\_size\_y** – height of the graph in cm
- **legendPosition** – position of the legend (NE, SE, SW, NW)

**Returns**

**class widget\_line\_drawer** (*minWinHeight=300, minWinWidth=300, is\_light=True*)

Bases: `PyQt5.QtWidgets.QWidget`

Widget allowing to display several lines easily

**signal\_must\_update**

**on\_update\_signal** (*self, listOfLines*)

**delete\_lines** (*self, key\_id*)

Dele the lines :param key\_id: id to delete :return:

**set\_lines** (*self, listOfLines, key\_id=0, pen=None*)

Set the lines to display :param listOfLines: list of [x1, y1, z1, x2, y2, z2] corresponding to lines :param key\_id: id of the trace :param pen: pen used to draw the lines :return:

**paintEvent** (*self, event, painter=None*)

**get\_extrema\_lines** (*self*)

**class widget\_menuButton** (*theParentButton*)

Bases: `PyQt5.QtWidgets.QMenu`

Same as QMenu, but integrates it behind a button more easily.

**showEvent** (*self, QShowEvent*)

**class widget\_openGL** (*parent=None*)

Bases: `PyQt5.QtWidgets.QOpenGLWidget`

Interface that provides opengl capabilities. Ensures zoom, light, rotation, etc.

**sizeHint** (*self*)

**minimumSizeHint** (*self*)

```
set_deviceDrawer (self, theDeviceDrawer)
    Set a drawer optimeed.visualize.gui.widgets.openglWidget.DeviceDrawerInterface.DeviceDrawerInterface

set_deviceToDraw (self, theDeviceToDraw)
    Set the device to draw optimeed.InterfaceDevice.InterfaceDevice

initializeGL (self)

paintGL (self)

resizeGL (self, w, h)

mousePressEvent (self, event)

mouseMoveEvent (self, event)

keyPressEvent (self, event)

wheelEvent (self, QWheelEvent)

class widget_text (theText, is_light=False, convertToHtml=False)
    Bases: PyQt5.QtWidgets.QLabel
    Widget able to display a text

    set_text (self, theText, convertToHtml=False)
        Set the text to display

class guiPyqtgraph (graphsVisual, **kwargs)
    Create a gui for pyqtgraph with trace selection options, export and action on clic choices

    refreshTraceList (self)
        Refresh all the traces

class DeviceDrawerInterface

    keyboard_push_action (self, theKey)

    get_colour_scalebar (self)

    get_colour_background (self)

    get_opengl_options (self)

class on_graph_click_delete (theDataLink)
    Bases: optimeed.visualize.gui.widgets.widget_graphs_visual.on_graph_click_interface
    On Click: Delete the points from the graph, and save the modified collection

    apply (self)

    reset (self)

    graph_clicked (self, theGraphVisual, index_graph, index_trace, indices_points)

    get_name (self)

class on_graph_click_export (theDataLink)
    Bases: optimeed.visualize.gui.widgets.widget_graphs_visual.on_graph_click_interface
    On click: export the selected points

    graph_clicked (self, theGraphVisual, index_graph, index_trace, indices_points)
```

```

    reset_graph (self)

    get_name (self)

class on_click_extract_pareto (theDataLink, max_x=False, max_y=False)
    Bases:                                     optimeed.visualize.gui.widgets.widget_graphs_visual.
                                                on_graph_click_interface
    On click: extract the pareto from the cloud of points

    graph_clicked (self, the_graph_visual, index_graph, index_trace, _)

    get_name (self)

class on_graph_click_showInfo (theLinkDataGraph, visuals=None)
    Bases:                                     optimeed.visualize.gui.widgets.widget_graphs_visual.
                                                on_graph_click_interface
    On click: show informations about the points (loop through attributes)

class DataInformationVisuals

    delete_visual (self, theVisual)

    add_visual (self, theVisual, theTrace, indexPoint)

    get_new_index (self)

    curr_index (self)

    graph_clicked (self, 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 (self)

class Repr_opengl (DeviceDrawer)

    get_widget (self, theNewDevice)

class Repr_lines (attribute_lines)

    get_widget (self, theNewDevice)

class on_graph_click_remove_trace (theDataLink)
    Bases:                                     optimeed.visualize.gui.widgets.widget_graphs_visual.
                                                on_graph_click_interface
    graph_clicked (self, theGraphVisual, index_graph, index_trace, _)

    get_name (self)

class on_click_copy_something (theDataLink, functionStrFromDevice)
    Bases:                                     optimeed.visualize.gui.widgets.widget_graphs_visual.
                                                on_graph_click_interface
    On Click: copy something

    graph_clicked (self, the_graph_visual, index_graph, index_trace, indices_points)

    get_name (self)

```

```
class on_click_change_symbol (theLinkDataGraph)
    Bases: optimeed.visualize.gui.widgets.widget_graphs_visual.
            on_graph_click_interface

    On Click: Change the symbol of the point that is clicked

    graph_clicked (self, theGraphVisual, index_graph, index_trace, indices_points)

    get_name (self)

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

class DataAnimationVisuals (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 (self, 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 (self, trace_id, indexPoint)

    delete_point (self, trace_id, thePoint)

    reset_all (self)

    delete_all (self)

    pause_play (self)

    show_all (self)

    next_frame (self)

    slider_handler (self)

    frame_selector (self)

    set_refreshTime (self)

    is_empty (self)

    run (self)

    closeEvent (self, _)

    contains_trace (self, trace_id)

    export_picture (self)
```

```

class DataAnimationOpenGL (theOpenGLWidget, theId=0, window_title='Animation')
    Bases: optimeed.visualize.gui.gui_data_animation.DataAnimationVisuals
    Implements DataAnimationVisuals to show opengl drawing
    update_widget_w_animation (self, key, index, the_data_animation)
    export_widget (self, painter)
    delete_key_widgets (self, key)

class DataAnimationOpenGLwText (*args, is_light=True, **kwargs)
    Bases: optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick.on_click_anim.DataAnimationOpenGL
    Implements DataAnimationVisuals to show opengl drawing and text
    update_widget_w_animation (self, key, index, the_data_animation)
    get_interesting_elements (self, devices_list)

class DataAnimationLines (is_light=True, theId=0, window_title='Animation')
    Bases: optimeed.visualize.gui.gui_data_animation.DataAnimationVisuals
    Implements DataAnimationVisuals to show drawing made out of lines (widget_line_drawer)
    export_widget (self, painter)
    delete_key_widgets (self, key)
    update_widget_w_animation (self, key, index, the_data_animation)
    get_interesting_elements (self, devices_list)

class DataAnimationVisualswText (is_light=True, theId=0, window_title='Animation')
    Bases: optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick.on_click_anim.DataAnimationLines
    Same as DataAnimationLines but also with text
    update_widget_w_animation (self, key, index, the_data_animation)

class on_graph_click_showAnim (theLinkDataGraph, theAnimation)
    Bases: optimeed.visualize.gui.widgets.widget_graphs_visual.on_graph_click_interface
    On click: add or remove an element to animate
    graph_clicked (self, theGraphVisual, index_graph, index_trace, indices_points)
    get_name (self)

```

## displayOptimization

### Module Contents

```

class OptimizationDisplayer (thePipeOpti, listOfObjectives, theOptimizer, additionalWid-
                             gets=None)
    Class used to display optimization process in real time
    signal_optimization_over
    set_actionsOnClick (self, theList)
        Set actions to perform on click, list of on_graph_click_interface

```

```
generate_optimizationGraphs (self, refresh_time=0.1)
    Generates the optimization graphs. :return: Graphs, LinkDataGraph,
    :class:'~optimeed.visulaize.gui.widgets.widget_graphs_visual.widget_graphs_visual

create_main_window (self)
    From the widgets and the actions on click, spawn a window and put a gui around widgetsGraphsVisual.

__change_appearance_violate_constraints (self)

__auto_refresh (self, refresh_time)

__set_graphs_disposition (self)
    Set nicely the graphs disposition

launch_optimization (self)
    Perform the optimization and spawn the convergence graphs afterwards.

__callback_optimization (self, myWindow)

class Worker
    Bases: PyQt5.QtCore.QObject

    signal_show_UI

    display_graphs (self, theGraphs)
```

*fastPlot*

## Module Contents

### **class PlotHolders**

```
add_plot (self, x, y, **kwargs)

get_wgGraphs (self)

new_plot (self)

set_title (self, theTitle, **kwargs)

reset (self)

axis_equal (self)
```

### **class WindowHolders**

```
set_currFigure (self, currFigure)

add_plot (self, *args, **kwargs)

set_title (self, *args, **kwargs)

new_figure (self)

new_plot (self)

show (self)

get_curr_plotHolder (self)

get_wgGraphs (self, fig=None)

get_all_figures (self)
```



```

    axis_equal (self)
myWindows
plot (x, y, hold=False, **kwargs)
    Plot new trace
show ()
    Show (start qt mainloop) graphs. Blocking
figure (numb)
    Set current figure
new_plot ()
    Add new plot
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

```

## Package Contents

```

class gui_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 (self, actionOnWindowClosed)

    closeEvent (self, event)

    run (self, hold=False)
        Display the window

    keyPressEvent (self, event)

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

class gui_collection_exporter
    Bases: PyQt5.QtWidgets.QMainWindow

    Simple gui that allows to export data

    signal_has_exported

    signal_has_reset

    exportCollection (self)
        Export the collection

    reset (self)

```

**add\_data\_to\_collection** (*self*, *data*)

Add data to the collection to export

**Parameters** *data* – Whichever type you like

**set\_info** (*self*, *info*)

**set\_collection** (*self*, *theCollection*)

**class DataAnimationVisuals** (*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** (*self*, *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** (*self*, *trace\_id*, *indexPoint*)

**delete\_point** (*self*, *trace\_id*, *thePoint*)

**reset\_all** (*self*)

**delete\_all** (*self*)

**pause\_play** (*self*)

**show\_all** (*self*)

**next\_frame** (*self*)

**slider\_handler** (*self*)

**frame\_selector** (*self*)

**set\_refreshTime** (*self*)

**is\_empty** (*self*)

**run** (*self*)

**closeEvent** (*self*, *\_*)

**contains\_trace** (*self*, *trace\_id*)

**export\_picture** (*self*)

**class widget\_graphs\_visual** (*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** (*self*, *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** (*self*, *idGraph*)

**\_\_check\_graphs** (*self*)

**on\_click** (*self*, *plotDataItem*, *clicked\_points*)

**update\_graphs** (*self*, *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** (*self*)

Use this method to update the graph in a fast way. NOT THREAD SAFE.

**exportGraphs** (*self*)

Export the graphs

**link\_axes** (*self*)

**get\_graph** (*self*, *idGraph*)

Get corresponding `GraphVisual` of the graph `idGraph`

**keyPressEvent** (*self*, *event*)

What happens if a key is pressed. R: reset the axes to their default value

**delete\_graph** (*self*, *idGraph*)

Delete the graph `idGraph`

**delete** (*self*)

**get\_all\_graphsVisual** (*self*)

Return a dictionary {`idGraph`: `GraphVisual`}.

**get\_layout\_buttons** (*self*)

Get the `QGraphicsLayout` where it's possible to add buttons, etc.

**set\_actionOnClick** (*self*, *theActionOnClick*)

Action to perform when the graph is clicked

**Parameters theActionOnClick** – `on_graph_click_interface`

**Returns**

**set\_title** (*self*, *idGraph*, *titleName*, *\*\*kwargs*)

Set title of the graph

**Parameters**

- **idGraph** – id of the graph
- **titleName** – title to set

**set\_article\_template** (*self*, *graph\_size\_x*=8.8, *graph\_size\_y*=4.4, *legendPosition*='NW')

Method to set the graphs to article quality graph.

**Parameters**

- **graph\_size\_x** – width of the graph in cm
- **graph\_size\_y** – height of the graph in cm
- **legendPosition** – position of the legend (NE, SE, SW, NW)

**Returns**

**class widget\_line\_drawer** (*minWinHeight*=300, *minWinWidth*=300, *is\_light*=True)

Bases: `PyQt5.QtWidgets.QWidget`

Widget allowing to display several lines easily

**signal\_must\_update**

**on\_update\_signal** (*self*, *listOfLines*)

**delete\_lines** (*self*, *key\_id*)

Dele the lines :param *key\_id*: id to delete :return:

**set\_lines** (*self*, *listOfLines*, *key\_id*=0, *pen*=None)

Set the lines to display :param *listOfLines*: list of [*x1*, *y1*, *z1*, *x2*, *y2*, *z2*] corresponding to lines :param *key\_id*: id of the trace :param *pen*: pen used to draw the lines :return:

**paintEvent** (*self*, *event*, *painter*=None)

**get\_extrema\_lines** (*self*)

**class widget\_menuButton** (*theParentButton*)

Bases: `PyQt5.QtWidgets.QMenu`

Same as `QMenu`, but integrates it behind a button more easily.

**showEvent** (*self*, *QShowEvent*)

**class widget\_openGL** (*parent*=None)

Bases: `PyQt5.QtWidgets.QOpenGLWidget`

Interface that provides opengl capabilities. Ensures zoom, light, rotation, etc.

**sizeHint** (*self*)

**minimumSizeHint** (*self*)

**set\_deviceDrawer** (*self*, *theDeviceDrawer*)

Set a drawer `optimeed.visualize.gui.widgets.openGLWidget.DeviceDrawerInterface.DeviceDrawerInterface`

**set\_deviceToDraw** (*self*, *theDeviceToDraw*)

Set the device to draw `optimeed.InterfaceDevice.InterfaceDevice`

**initializeGL** (*self*)

**paintGL** (*self*)

**resizeGL** (*self*, *w*, *h*)

```

    mousePressEvent (self, event)

    mouseMoveEvent (self, event)

    keyPressEvent (self, event)

    wheelEvent (self, QWheelEvent)

class widget_text (theText, is_light=False, convertToHtml=False)
    Bases: PyQt5.QtWidgets.QLabel
    Widget able to display a text

    set_text (self, theText, convertToHtml=False)
        Set the text to display

class guiPyqtgraph (graphsVisual, **kwargs)
    Create a gui for pyqtgraph with trace selection options, export and action on clic choices

    refreshTraceList (self)
        Refresh all the traces

class DeviceDrawerInterface

    keyboard_push_action (self, theKey)

    get_colour_scalebar (self)

    get_colour_background (self)

    get_opengl_options (self)

class on_graph_click_delete (theDataLink)
    Bases: optimeed.visualize.gui.widgets.widget_graphs_visual.on_graph_click_interface
    On Click: Delete the points from the graph, and save the modified collection

    apply (self)

    reset (self)

    graph_clicked (self, theGraphVisual, index_graph, index_trace, indices_points)

    get_name (self)

class on_graph_click_export (theDataLink)
    Bases: optimeed.visualize.gui.widgets.widget_graphs_visual.on_graph_click_interface
    On click: export the selected points

    graph_clicked (self, theGraphVisual, index_graph, index_trace, indices_points)

    reset_graph (self)

    get_name (self)

class on_click_extract_pareto (theDataLink, max_x=False, max_y=False)
    Bases: optimeed.visualize.gui.widgets.widget_graphs_visual.on_graph_click_interface
    On click: extract the pareto from the cloud of points

    graph_clicked (self, the_graph_visual, index_graph, index_trace, _)

    get_name (self)

```

```
class on_graph_click_showInfo (theLinkDataGraph, visuals=None)
    Bases: optimeed.visualize.gui.widgets.widget_graphs_visual.
            on_graph_click_interface

    On click: show informations about the points (loop through attributes)

    class DataInformationVisuals

        delete_visual (self, theVisual)

        add_visual (self, theVisual, theTrace, indexPoint)

        get_new_index (self)

        curr_index (self)

        graph_clicked (self, 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 (self)

class Repr_opengl (DeviceDrawer)

    get_widget (self, theNewDevice)

class Repr_lines (attribute_lines)

    get_widget (self, theNewDevice)

class on_graph_click_remove_trace (theDataLink)
    Bases: optimeed.visualize.gui.widgets.widget_graphs_visual.
            on_graph_click_interface

    graph_clicked (self, theGraphVisual, index_graph, index_trace, _)

    get_name (self)

class on_click_copy_something (theDataLink, functionStrFromDevice)
    Bases: optimeed.visualize.gui.widgets.widget_graphs_visual.
            on_graph_click_interface

    On Click: copy something

    graph_clicked (self, the_graph_visual, index_graph, index_trace, indices_points)

    get_name (self)

class on_click_change_symbol (theLinkDataGraph)
    Bases: optimeed.visualize.gui.widgets.widget_graphs_visual.
            on_graph_click_interface

    On Click: Change the symbol of the point that is clicked

    graph_clicked (self, theGraphVisual, index_graph, index_trace, indices_points)

    get_name (self)

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

```

class DataAnimationOpenGL (theOpenGLWidget, theId=0, window_title='Animation')
    Bases: optimeed.visualize.gui.gui_data_animation.DataAnimationVisuals
    Implements DataAnimationVisuals to show opengl drawing
    update_widget_w_animation (self, key, index, the_data_animation)
    export_widget (self, painter)
    delete_key_widgets (self, key)

class DataAnimationOpenGLwText (*args, is_light=True, **kwargs)
    Bases: optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick.on_click_anim.DataAnimationOpenGL
    Implements DataAnimationVisuals to show opengl drawing and text
    update_widget_w_animation (self, key, index, the_data_animation)
    get_interesting_elements (self, devices_list)

class DataAnimationLines (is_light=True, theId=0, window_title='Animation')
    Bases: optimeed.visualize.gui.gui_data_animation.DataAnimationVisuals
    Implements DataAnimationVisuals to show drawing made out of lines (widget_line_drawer)
    export_widget (self, painter)
    delete_key_widgets (self, key)
    update_widget_w_animation (self, key, index, the_data_animation)
    get_interesting_elements (self, devices_list)

class DataAnimationVisualswText (is_light=True, theId=0, window_title='Animation')
    Bases: optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick.on_click_anim.DataAnimationLines
    Same as DataAnimationLines but also with text
    update_widget_w_animation (self, key, index, the_data_animation)

class on_graph_click_showAnim (theLinkDataGraph, theAnimation)
    Bases: optimeed.visualize.gui.widgets.widget_graphs_visual.on_graph_click_interface
    On click: add or remove an element to animate
    graph_clicked (self, theGraphVisual, index_graph, index_trace, indices_points)
    get_name (self)

class LinkDataGraph

    class _collection_linker

        add_link (self, idSlave, idMaster)
        get_collection_master (self, idToGet)
        is_slave (self, idToCheck)
        set_same_master (self, idExistingSlave, idOtherSlave)
        Parameters
        • idExistingSlave – id collection of the existing slave

```

- **idOtherSlave** – id collection of the new slave that has to be linked to an existing master

```
add_collection (self, theCollection, kwargs=None)  
add_graph (self, howToPlotGraph)  
createGraphs (self)  
get_howToPlotGraph (self, idGraph)  
get_collectionInfo (self, idCollectionInfo)  
create_trace (self, collectionInfo, howToPlotGraph, idGraph)  
get_all_id_graphs (self)  
get_all_traces_id_graph (self, idGraph)  
update_graphs (self)  
is_slave (self, idGraph, idTrace)  
get_idCollection_from_graph (self, idGraph, idTrace, getMaster=True)  
    From indices in the graph, get index of corresponding collection  
get_collection_from_graph (self, idGraph, idTrace, getMaster=True)  
    From indices in the graph, get corresponding collection  
get_dataObject_from_graph (self, idGraph, idTrace, idPoint)  
get_dataObjects_from_graph (self, idGraph, idTrace, idPoint_list)  
remove_element_from_graph (self, idGraph, idTrace, idPoint, deleteFromMaster=False)  
    Remove element from the graph, or the master collection  
remove_elements_from_trace (self, idGraph, idTrace, idPoints, deleteFromMaster=False)  
    Performances      optimisation      when      compared      to      LinkDataGraph.  
    remove_element_from_graph()  
link_collection_to_graph_collection (self, id_collection_graph, id_collection_master)  
    Link data :param id_collection_graph: :param id_collection_master: :return:  
remove_trace (self, idGraph, idTrace)  
get_graph_and_trace_from_collection (self, idCollection)  
    Reverse search: from a collection, get the associated graph  
get_mappingData_graph (self, idGraph)  
get_mappingData_trace (self, idGraph, idTrace)  
class HowToPlotGraph (attribute_x, attribute_y, kwargs_graph=None, excluded=None)  
  
    exclude_col (self, id_col)  
        Add id_col to exclude from the graph  
    __str__ (self)  
  
class gui_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 (self, actionOnWindowClosed)
```



**closeEvent** (*self*, *event*)

**run** (*self*, *hold=False*)  
Display the window

**keyPressEvent** (*self*, *event*)

**class widget\_graphs\_visual** (*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** (*self*, *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** (*self*, *idGraph*)

**\_\_check\_graphs** (*self*)

**on\_click** (*self*, *plotDataItem*, *clicked\_points*)

**update\_graphs** (*self*, *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 refresh each `self.refreshTime`

**fast\_update** (*self*)

Use this method to update the graph in a fast way. NOT THREAD SAFE.

**exportGraphs** (*self*)

Export the graphs

**link\_axes** (*self*)

**get\_graph** (*self*, *idGraph*)

Get corresponding `GraphVisual` of the graph `idGraph`

**keyPressEvent** (*self*, *event*)

What happens if a key is pressed. R: reset the axes to their default value

**delete\_graph** (*self*, *idGraph*)

Delete the graph `idGraph`

**delete** (*self*)

**get\_all\_graphsVisual** (*self*)

Return a dictionary {idGraph: GraphVisual}.

**get\_layout\_buttons** (*self*)

Get the QGraphicsLayout where it's possible to add buttons, etc.

**set\_actionOnClick** (*self*, *theActionOnClick*)

Action to perform when the graph is clicked

**Parameters** **theActionOnClick** – *on\_graph\_click\_interface*

**Returns**

**set\_title** (*self*, *idGraph*, *titleName*, *\*\*kwargs*)

Set title of the graph

**Parameters**

- **idGraph** – id of the graph
- **titleName** – title to set

**set\_article\_template** (*self*, *graph\_size\_x*=8.8, *graph\_size\_y*=4.4, *legendPosition*='NW')

Method to set the graphs to article quality graph.

**Parameters**

- **graph\_size\_x** – width of the graph in cm
- **graph\_size\_y** – height of the graph in cm
- **legendPosition** – position of the legend (NE, SE, SW, NW)

**Returns**

**class on\_graph\_click\_showInfo** (*theLinkDataGraph*, *visuals*=None)

Bases: *optimeed.visualize.gui.widgets.widget\_graphs\_visual.on\_graph\_click\_interface*

On click: show informations about the points (loop through attributes)

**class DataInformationVisuals**

**delete\_visual** (*self*, *theVisual*)

**add\_visual** (*self*, *theVisual*, *theTrace*, *indexPoint*)

**get\_new\_index** (*self*)

**curr\_index** (*self*)

**graph\_clicked** (*self*, *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** (*self*)

**class guiPyqtgraph** (*graphsVisual*, *\*\*kwargs*)

Create a gui for pyqtgraph with trace selection options, export and action on clic choices

**refreshTraceList** (*self*)

Refresh all the traces

**class OptimizationDisplayer** (*thePipeOpti*, *listOfObjectives*, *theOptimizer*, *additionalWid-*  
*gets*=None)

Class used to display optimization process in real time

```

signal_optimization_over

set_actionsOnClick (self, theList)
    Set actions to perform on click, list of on_graph_click_interface

generate_optimizationGraphs (self, refresh_time=0.1)
    Generates the optimization graphs. :return: Graphs, LinkDataGraph,
    :class: '~optimeed.visulaize.gui.widgets.widget_graphs_visual.widget_graphs_visual'

create_main_window (self)
    From the widgets and the actions on click, spawn a window and put a gui around widgetGraphsVisual.

__change_appearance_violate_constraints (self)

__auto_refresh (self, refresh_time)

__set_graphs_disposition (self)
    Set nicely the graphs disposition

launch_optimization (self)
    Perform the optimization and spawn the convergence graphs afterwards.

__callback_optimization (self, myWindow)

class Worker
    Bases: PyQt5.QtCore.QObject

    signal_show_UI

    display_graphs (self, theGraphs)

class widget_graphs_visual (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 (self, 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 (self, idGraph)

    __check_graphs (self)

    on_click (self, plotDataItem, clicked_points)

    update_graphs (self, 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 refresh each `self.refreshTime`

**fast\_update** (*self*)

Use this method to update the graph in a fast way. NOT THREAD SAFE.

**exportGraphs** (*self*)

Export the graphs

**link\_axes** (*self*)

**get\_graph** (*self*, *idGraph*)

Get corresponding `GraphVisual` of the graph `idGraph`

**keyPressEvent** (*self*, *event*)

What happens if a key is pressed. R: reset the axes to their default value

**delete\_graph** (*self*, *idGraph*)

Delete the graph `idGraph`

**delete** (*self*)

**get\_all\_graphsVisual** (*self*)

Return a dictionary {`idGraph`: `GraphVisual`}.

**get\_layout\_buttons** (*self*)

Get the `QGraphicsLayout` where it's possible to add buttons, etc.

**set\_actionOnClick** (*self*, *theActionOnClick*)

Action to perform when the graph is clicked

**Parameters** `theActionOnClick` – *on\_graph\_click\_interface*

**Returns**

**set\_title** (*self*, *idGraph*, *titleName*, *\*\*kwargs*)

Set title of the graph

**Parameters**

- **idGraph** – id of the graph
- **titleName** – title to set

**set\_article\_template** (*self*, *graph\_size\_x*=8.8, *graph\_size\_y*=4.4, *legendPosition*='NW')

Method to set the graphs to article quality graph.

**Parameters**

- **graph\_size\_x** – width of the graph in cm
- **graph\_size\_y** – height of the graph in cm
- **legendPosition** – position of the legend (NE, SE, SW, NW)

**Returns**

**class** `gui_mainWindow` (`QtWidgetList`, *isLight*=True, *actionOnWindowClosed*=None, *neverCloseWindow*=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** (*self*, *actionOnWindowClosed*)

**closeEvent** (*self*, *event*)

---

```

run (self, hold=False)
    Display the window

keyPressEvent (self, event)

start_qt_mainloop ()
    Starts qt mainloop, which is necessary for qt to handle events

stop_qt_mainloop ()
    Stops qt mainloop and resumes to program

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, symbol='o', symbolsize=8, fillsymbol=True, outlinesym-
    bol=1.8, linestyle='-', width=2)
    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_data (self, x: list, y: list)
    Overwrites current datapoints with new set

get_x (self)
    Get x coordinates of datapoints

get_symbolsize (self)
    Get size of the symbols

symbol_isfilled (self)
    Check if symbols has to be filled or not

get_symbolOutline (self)
    Get color factor of outline of symbols

get_length_data (self)
    Get number of points

get_xlim (self)
    Get x limits of viewbox

get_ylim (self)
    Get y limits of viewbox

get_y (self)
    Get y coordinates of datapoints

get_color (self)
    Get color of the line

get_width (self)
    Get width of the line

get_number_of_points (self)
    Get number of points

get_plot_data (self)
    Call this method to get the x and y coordinates of the points that have to be displayed. => After transfor-
    mation, and after permutations.

    Returns x (list), y (list)

get_permutations (self)
    Return the transformation 'permutation': xplot[i] = xdata[permutation[i]]

```

**get\_invert\_permutations** (*self*)

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

**get\_dataIndex\_from\_graphIndex** (*self*, *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** (*self*, *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** (*self*, *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** (*self*, *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** (*self*, *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** (*self*)

Get x label of the trace

**get\_y\_label** (*self*)

Get y label of the trace

**get\_legend** (*self*)

Get name of the trace

**get\_symbol** (*self*)

Get symbol

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

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

**delete\_point** (*self*, *index\_point*)

Delete a point from the datapoints

**is\_scattered** (*self*)

Delete a point from the datapoints

**set\_indices\_points\_to\_plot** (*self*, *indices*)

Set indices points to plot

**get\_indices\_points\_to\_plot** (*self*)

Get indices points to plot

**get\_linestyle** (*self*)

Get linestyle

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

**export\_str** (*self*)

Method to save the points constituting the trace

**class Graphs**

Contains several Graph

**updateChildren** (*self*)

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

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

**add\_trace** (*self*, *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** (*self*, *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** (*self*)

Get id of the first graph

**Returns** id of the first graph

**get\_graph** (*self*, *idGraph*)

Get graph object at idgraph

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

**Returns** Graph

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

Get all ids of the graphs

**Returns** list of id graphs

**get\_all\_graphs** (*self*)

Get all graphs. Return dict {id: Graph}

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

Add a new graph

**Returns** id of the created graph

**remove\_graph** (*self*, *idGraph*)

Delete a graph

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

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

Add a callback each time a graph is modified.

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

**export\_str** (*self*)

Export all the graphs in text

**Returns** `str`

**merge** (*self*, *otherGraphs*)

**reset** (*self*)

**class** `guiPyqtgraph` (*graphsVisual*, *\*\*kwargs*)

Create a gui for pyqtgraph with trace selection options, export and action on clic choices

**refreshTraceList** (*self*)

Refresh all the traces

**class** `PlotHolders`

**add\_plot** (*self*, *x*, *y*, *\*\*kwargs*)

**get\_wgGraphs** (*self*)

**new\_plot** (*self*)

**set\_title** (*self*, *theTitle*, *\*\*kwargs*)

**reset** (*self*)

**axis\_equal** (*self*)

**class** `WindowHolders`

**set\_currFigure** (*self*, *currFigure*)

**add\_plot** (*self*, *\*args*, *\*\*kwargs*)

**set\_title** (*self*, *\*args*, *\*\*kwargs*)

**new\_figure** (*self*)

**new\_plot** (*self*)

**show** (*self*)

**get\_curr\_plotHolder** (*self*)

**get\_wgGraphs** (*self*, *fig=None*)

**get\_all\_figures** (*self*)

**axis\_equal** (*self*)

**myWindows**

**plot** (*x*, *y*, *hold=False*, *\*\*kwargs*)

Plot new trace

**show** ()

Show (start qt mainloop) graphs. Blocking



**figure** (*numb*)  
Set current figure

**new\_plot** ()  
Add new plot

**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*

### 6.1.2 Package Contents

**VERSION = 1.0.2**



## 7.1 documentation

### 7.1.1 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/>

### 7.1.2 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::*python setup.py upload*



### O

[optimeed](#), 17  
[optimeed consolidate](#), 17  
[optimeed consolidate.parametric\\_analysis](#), 17  
[optimeed core](#), 19  
[optimeed core ansi2html](#), 19  
[optimeed core ansi2html.converter](#), 19  
[optimeed core ansi2html.style](#), 21  
[optimeed core ansi2html.util](#), 21  
[optimeed core collection](#), 22  
[optimeed core color\\_palette](#), 24  
[optimeed core commonImport](#), 24  
[optimeed core graphs](#), 24  
[optimeed core interfaceDevice](#), 28  
[optimeed core linkDataGraph](#), 28  
[optimeed core myjson](#), 29  
[optimeed core options](#), 30  
[optimeed core tools](#), 30  
[optimeed optimize](#), 43  
[optimeed optimize characterization](#), 43  
[optimeed optimize characterization.characterization](#), 43  
[optimeed optimize characterization.interfaceCharacterization](#), 44  
[optimeed optimize mathsToPhysics](#), 44  
[optimeed optimize mathsToPhysics.interfaceMathsToPhysics](#), 44  
[optimeed optimize mathsToPhysics.mathsToPhysics](#), 44  
[optimeed optimize objAndCons](#), 45  
[optimeed optimize objAndCons.fastObjCons](#), 45  
[optimeed optimize objAndCons.interfaceObjCons](#), 45  
[optimeed optimize optiAlgorithms](#), 46  
[optimeed optimize optiAlgorithms.algorithmInterface](#), 49  
[optimeed optimize optiAlgorithms.convergence](#), 46  
[optimeed optimize optiAlgorithms.convergence.evolution](#), 46  
[optimeed optimize optiAlgorithms.convergence.hyperparameter](#), 47  
[optimeed optimize optiAlgorithms.convergence.intermediate](#), 48  
[optimeed optimize optiAlgorithms.multiObjective\\_GA](#), 49  
[optimeed optimize optiAlgorithms.NLOpt\\_Algorithm](#), 48  
[optimeed optimize optimizer](#), 52  
[optimeed optimize optiVariable](#), 51  
[optimeed visualize](#), 56  
[optimeed visualize.displayOptimization](#), 91  
[optimeed visualize.fastPlot](#), 92  
[optimeed visualize.gui](#), 56  
[optimeed visualize.gui.gui\\_collection\\_exporter](#), 81  
[optimeed visualize.gui.gui\\_data\\_animation](#), 81  
[optimeed visualize.gui.gui\\_data\\_selector](#), 83  
[optimeed visualize.gui.gui\\_mainWindow](#), 84  
[optimeed visualize.gui.widgets](#), 56  
[optimeed visualize.gui.widgets.graphsVisualWidget](#), 56  
[optimeed visualize.gui.widgets.graphsVisualWidget.collection](#), 56  
[optimeed visualize.gui.widgets.graphsVisualWidget.collection](#), 56  
[optimeed visualize.gui.widgets.graphsVisualWidget.collection](#), 57  
[optimeed visualize.gui.widgets.graphsVisualWidget.collection](#), 58  
[optimeed visualize.gui.widgets.graphsVisualWidget.collection](#), 58  
[optimeed visualize.gui.widgets.graphsVisualWidget.collection](#), 58

58  
optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick.on\_click\_extract\_p  
59  
optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick.on\_click\_remove\_tr  
59  
optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick.on\_click\_showinfo,  
59  
optimeed.visualize.gui.widgets.graphsVisualWidget.graphVisual,  
63  
optimeed.visualize.gui.widgets.graphsVisualWidget.pyqtgraphRedefine,  
65  
optimeed.visualize.gui.widgets.graphsVisualWidget.smallGui,  
67  
optimeed.visualize.gui.widgets.graphsVisualWidget.traceVisual,  
67  
optimeed.visualize.gui.widgets.openGLWidget,  
69  
optimeed.visualize.gui.widgets.openGLWidget.ContextHandler,  
69  
optimeed.visualize.gui.widgets.openGLWidget.DeviceDrawerInterface,  
70  
optimeed.visualize.gui.widgets.openGLWidget.Materials\_visual,  
70  
optimeed.visualize.gui.widgets.openGLWidget.OpenGlFunctions\_Library,  
71  
optimeed.visualize.gui.widgets.openGLWidget.quaternions,  
72  
optimeed.visualize.gui.widgets.openGLWidget.TriangulatePolygon,  
71  
optimeed.visualize.gui.widgets.widget\_graphs\_visual,  
72  
optimeed.visualize.gui.widgets.widget\_line\_drawer,  
73  
optimeed.visualize.gui.widgets.widget\_menuButton,  
74  
optimeed.visualize.gui.widgets.widget\_openGL,  
74  
optimeed.visualize.gui.widgets.widget\_text,  
75

## Symbols

- `__CONVERGENCE` (*OptiHistoric* attribute), 52
  - `__DATA_STR` (*ListDataStruct* attribute), 23, 34
  - `__DEVICE` (*OptiHistoric* attribute), 52
  - `__INFO_STR` (*ListDataStruct* attribute), 23, 34
  - `__LOGOPTI` (*OptiHistoric* attribute), 52
  - `__RESULTS` (*OptiHistoric* attribute), 52
  - `__State` (class in *optimeed.core.ansi2html.converter*), 20
  - `__amb3__` (*MaterialRenderingProperties* attribute), 70
  - `__author__` (in module *optimeed.optimize.optiAlgorithms.convergence.hypervolume*), 47
  - `__auto_refresh()` (*OptimizationDisplayer* method), 92, 103
  - `__callback_optimization()` (*OptimizationDisplayer* method), 92, 103
  - `__change_appearance_violate_constraints()` (*OptimizationDisplayer* method), 92, 103
  - `__check_graphs()` (*widget\_graphs\_visual* method), 72, 75, 86, 95, 101, 103
  - `__create_graph()` (*widget\_graphs\_visual* method), 72, 75, 86, 95, 101, 103
  - `__dif3__` (*MaterialRenderingProperties* attribute), 70
  - `__draw_axis__()` (*ContextHandler* method), 69
  - `__len__()` (*MultiList* method), 47
  - `__lightingInit__()` (*ContextHandler* method), 69
  - `__reset__()` (*ContextHandler* method), 69
  - `__set_graphs_disposition()` (*OptimizationDisplayer* method), 92, 103
  - `__shin__` (*MaterialRenderingProperties* attribute), 70
  - `__spec3__` (*MaterialRenderingProperties* attribute), 70
  - `__str__()` (*Attribute\_selector* method), 83
  - `__str__()` (*AutosaveStruct* method), 22, 34
  - `__str__()` (*Binary\_OptimizationVariable* method), 51, 55
  - `__str__()` (*Container\_attribute\_selector* method), 83
  - `__str__()` (*Data* method), 26, 39, 107
  - `__str__()` (*DataStruct\_Interface* method), 22, 34
  - `__str__()` (*HowToPlotGraph* method), 28, 41, 100
  - `__str__()` (*Integer\_OptimizationVariable* method), 51, 55
  - `__str__()` (*InterfaceCharacterization* method), 44, 53
  - `__str__()` (*InterfaceObjCons* method), 45, 46, 54
  - `__str__()` (*MathsToPhysics* method), 44, 45, 54
  - `__str__()` (*MultiList* method), 47
  - `__str__()` (*MultiList.Node* method), 47
  - `__str__()` (*MultiObjective\_GA* method), 50, 54
  - `__str__()` (*NLOpt\_Algorithm* method), 49
  - `__str__()` (*OptimizationVariable* method), 51
  - `__str__()` (*Options* method), 30, 43
  - `__str__()` (*Real\_OptimizationVariable* method), 51, 55
  - `__str__()` (*Rule* method), 21
  - `__apply_regex()` (*Ansi2HTMLConverter* method), 21, 22
  - `__collapse_cursor()` (*Ansi2HTMLConverter* method), 21, 22
  - `__find_class()` (in module *optimeed.core.myjson*), 29
  - `__get_object_class()` (in module *optimeed.core.myjson*), 29
  - `__get_object_module()` (in module *optimeed.core.myjson*), 29
  - `__html_template` (in module *optimeed.core.ansi2html.converter*), 20
  - `__latex_template` (in module *optimeed.core.ansi2html.converter*), 20
  - `__needs_extra_newline()` (in module *optimeed.core.ansi2html.converter*), 20
  - `__object_to_FQCN()` (in module *optimeed.core.myjson*), 29
  - `__remove_index_from_show()` (*DataAnimation-Trace* method), 82
- ## A
- `Action_on_selector_update` (class in *optimeed.visualize.gui.gui\_data\_selector*), 83
  - `activateMaterialProperties()` (*MaterialRenderingProperties* method), 70

add\_attribute\_selector() (Container\_attribute\_selector method), 83  
 add\_child() (Attribute\_selector method), 83  
 add\_child() (Container\_attribute\_selector method), 83  
 add\_collection() (LinkDataGraph method), 28, 42, 100  
 add\_data() (GraphVisual method), 64  
 add\_data() (ListDataStruct method), 23, 34  
 add\_data\_to\_collection() (gui\_collection\_exporter method), 81, 85, 94  
 add\_element() (DataAnimationTrace method), 82  
 add\_elementToTrace() (DataAnimationVisuals method), 61, 79, 82, 85, 90, 94  
 add\_feature() (GraphVisual method), 64  
 add\_graph() (Graphs method), 27, 40, 107  
 add\_graph() (LinkDataGraph method), 28, 42, 100  
 add\_index\_to\_show() (DataAnimationTrace method), 82  
 add\_link() (LinkDataGraph\_collection\_linker method), 28, 41, 99  
 add\_modified\_paintElem() (TraceVisual.\_ModifiedPaintElem method), 67  
 add\_option() (Option\_class method), 18, 30, 43  
 add\_option() (Options method), 30, 43  
 add\_plot() (PlotHolders method), 92, 108  
 add\_plot() (WindowHolders method), 92, 108  
 add\_point() (ConvergenceManager method), 48  
 add\_point() (Data method), 26, 39, 106  
 add\_point() (OptiHistoric method), 52  
 add\_suffix\_to\_path() (in module optimeed.core), 36  
 add\_suffix\_to\_path() (in module optimeed.core.tools), 31  
 add\_trace() (DataAnimationVisuals method), 61, 79, 82, 85, 90, 94  
 add\_trace() (Graph method), 26, 39  
 add\_trace() (Graphs method), 27, 40, 107  
 add\_trace() (GraphVisual method), 64  
 add\_trace\_firstGraph() (Graphs method), 26, 40, 107  
 add\_update\_method() (Graphs method), 27, 41, 108  
 add\_visual() (on\_graph\_click\_showInfo.DataInformationVisuals method), 59, 60, 78, 89, 98, 102  
 addItem() (myGraphicsLayout method), 65  
 addItem() (myLegend method), 66  
 adjust() (\_State method), 20  
 ALGORITHM (NLOpt\_Algorithm attribute), 49  
 AlgorithmInterface (class in optimeed.optimize.optiAlgorithms.algorithmInterface), 49  
 Ansi2HTMLConverter (class in optimeed.core.ansi2html), 22  
 Ansi2HTMLConverter (class in optimeed.core.ansi2html.converter), 20  
 ANSI\_BACKGROUND\_256 (in module optimeed.core.ansi2html.converter), 20  
 ANSI\_BACKGROUND\_CUSTOM\_MAX (in module optimeed.core.ansi2html.converter), 20  
 ANSI\_BACKGROUND\_CUSTOM\_MIN (in module optimeed.core.ansi2html.converter), 20  
 ANSI\_BACKGROUND\_DEFAULT (in module optimeed.core.ansi2html.converter), 20  
 ANSI\_BACKGROUND\_HIGH\_INTENSITY\_MAX (in module optimeed.core.ansi2html.converter), 20  
 ANSI\_BACKGROUND\_HIGH\_INTENSITY\_MIN (in module optimeed.core.ansi2html.converter), 20  
 ANSI\_BLINK\_FAST (in module optimeed.core.ansi2html.converter), 19  
 ANSI\_BLINK\_OFF (in module optimeed.core.ansi2html.converter), 19  
 ANSI\_BLINK\_SLOW (in module optimeed.core.ansi2html.converter), 19  
 ANSI\_CROSSED\_OUT\_OFF (in module optimeed.core.ansi2html.converter), 19  
 ANSI\_CROSSED\_OUT\_ON (in module optimeed.core.ansi2html.converter), 19  
 ANSI\_FOREGROUND\_256 (in module optimeed.core.ansi2html.converter), 20  
 ANSI\_FOREGROUND\_CUSTOM\_MAX (in module optimeed.core.ansi2html.converter), 20  
 ANSI\_FOREGROUND\_CUSTOM\_MIN (in module optimeed.core.ansi2html.converter), 19  
 ANSI\_FOREGROUND\_DEFAULT (in module optimeed.core.ansi2html.converter), 20  
 ANSI\_FOREGROUND\_HIGH\_INTENSITY\_MAX (in module optimeed.core.ansi2html.converter), 20  
 ANSI\_FOREGROUND\_HIGH\_INTENSITY\_MIN (in module optimeed.core.ansi2html.converter), 20  
 ANSI\_FULL\_RESET (in module optimeed.core.ansi2html.converter), 19  
 ANSI\_INTENSITY\_INCREASED (in module optimeed.core.ansi2html.converter), 19  
 ANSI\_INTENSITY\_NORMAL (in module optimeed.core.ansi2html.converter), 19  
 ANSI\_INTENSITY\_REDUCED (in module optimeed.core.ansi2html.converter), 19  
 ANSI\_NEGATIVE\_OFF (in module optimeed.core.ansi2html.converter), 20  
 ANSI\_NEGATIVE\_ON (in module optimeed.core.ansi2html.converter), 20  
 ANSI\_STYLE\_ITALIC (in module optimeed.core.ansi2html.converter), 19  
 ANSI\_STYLE\_NORMAL (in module optimeed.core.ansi2html.converter), 19



- ANSI\_UNDERLINE\_OFF (in module *optimeed.core.ansi2html.converter*), 19
- ANSI\_UNDERLINE\_ON (in module *optimeed.core.ansi2html.converter*), 19
- ANSI\_VISIBILITY\_OFF (in module *optimeed.core.ansi2html.converter*), 19
- ANSI\_VISIBILITY\_ON (in module *optimeed.core.ansi2html.converter*), 19
- app (in module *optimeed.visualize*), 93
- app (in module *optimeed.visualize.gui*), 84
- app (in module *optimeed.visualize.gui.gui\_data\_selector*), 83
- app (in module *optimeed.visualize.gui.gui\_mainWindow*), 84
- append() (*MultiList* method), 47
- apply() (*on\_graph\_click\_delete* method), 58, 60, 77, 88, 97
- apply\_filters() (*GuiDataSelector* method), 83
- apply\_palette() (*GraphVisual* method), 64
- apply\_regex() (*Ansi2HTMLConverter* method), 21, 22
- apply\_width\_sample() (*myLegend* method), 66
- applyEquation() (in module *optimeed.core*), 33, 35
- applyEquation() (in module *optimeed.core.tools*), 31
- arithmeticEval() (in module *optimeed.core*), 36
- arithmeticEval() (in module *optimeed.core.tools*), 31
- assign() (*InterfaceDevice* method), 28, 41
- Attribute\_selector (class in *optimeed.visualize.gui.gui\_data\_selector*), 83
- attrs() (*Ansi2HTMLConverter* method), 21, 22
- AutosaveStruct (class in *optimeed.core*), 34
- AutosaveStruct (class in *optimeed.core.collection*), 22
- axis\_equal() (*GraphVisual* method), 65
- axis\_equal() (in module *optimeed.visualize*), 109
- axis\_equal() (in module *optimeed.visualize.fastPlot*), 93
- axis\_equal() (*PlotHolders* method), 92, 108
- axis\_equal() (*WindowHolders* method), 92, 108
- axisangle\_to\_q() (in module *optimeed.visualize.gui.widgets.openGLWidget.quaternions*), 72
- B**
- Bar (class in *optimeed.core.myjson*), 30
- Binary\_OptimizationVariable (class in *optimeed.optimize*), 55
- Binary\_OptimizationVariable (class in *optimeed.optimize.optiVariable*), 51
- blackOnly() (in module *optimeed.core.color\_palette*), 24
- BLUE (*text\_format* attribute), 18, 31, 35, 42
- Blue\_material (in module *optimeed.visualize.gui.widgets.openGLWidget.Materials\_visual*), 70
- BOLD (*text\_format* attribute), 18, 31, 35, 43
- Brass\_material (in module *optimeed.visualize.gui.widgets.openGLWidget.Materials\_visual*), 70
- Bronze\_material (in module *optimeed.visualize.gui.widgets.openGLWidget.Materials\_visual*), 70
- C**
- callback\_on\_evaluation() (*Optimizer* method), 53, 56
- cart2pol() (in module *optimeed.core*), 36
- cart2pol() (in module *optimeed.core.tools*), 31
- Characterization (class in *optimeed.optimize*), 54
- Characterization (class in *optimeed.optimize.characterization*), 44
- Characterization (class in *optimeed.optimize.characterization.characterization*), 43
- check\_and\_add\_if\_float() (in module *optimeed.visualize.gui.gui\_data\_selector*), 84
- Chrome\_material (in module *optimeed.visualize.gui.widgets.openGLWidget.Materials\_visual*), 70
- CLASS\_TAG (in module *optimeed.core.myjson*), 29
- CLIC\_LEFT (in module *optimeed.visualize.gui.widgets.openGLWidget.ContextHandler*), 69
- CLIC\_RIGHT (in module *optimeed.visualize.gui.widgets.openGLWidget.ContextHandler*), 69
- close() (*MyMultiprocessEvaluator* method), 50
- closeEvent() (*DataAnimationVisuals* method), 62, 79, 83, 85, 90, 94
- closeEvent() (*gui\_mainWindow* method), 84, 93, 100, 104
- CollectionInfo (class in *optimeed.core*), 41
- CollectionInfo (class in *optimeed.core.linkDataGraph*), 28
- color() (in module *optimeed.core.ansi2html.style*), 21
- color\_component() (in module *optimeed.core.ansi2html.style*), 21
- compute() (*Characterization* method), 43, 44, 54
- compute() (*FastObjCons* method), 45, 54
- compute() (*HyperVolume* method), 47
- compute() (*MultiObjective\_GA* method), 50, 54
- compute() (*NLOpt\_Algorithm* method), 49
- constraints (*OptiHistoric.\_pointData* attribute), 52
- constraints\_per\_step (*EvolutionaryConvergence* attribute), 46, 48

Container\_attribute\_selector (class in *optimeed.visualize.gui.gui\_data\_selector*), 83  
 contains\_trace() (*DataAnimationVisuals* method), 62, 79, 83, 86, 90, 94  
 ContextHandler (class in *optimeed.visualize.gui.widgets.openGLWidget.ContextHandler*), 69  
 conv (*MyConvergence* attribute), 49  
 ConvergenceManager (class in *optimeed.optimize.optiAlgorithms.NLOpt\_Algorithm*), 48  
 convert() (*Ansi2HTMLConverter* method), 21, 22  
 Copper\_material (in module *optimeed.visualize.gui.widgets.openGLWidget.Materials\_visual*), 70  
 copy() (*Options* method), 30, 43  
 create\_main\_window() (*OptimizationDisplayer* method), 92, 103  
 create\_trace() (*LinkDataGraph* method), 29, 42, 100  
 create\_unique\_dirname() (in module *optimeed.core*), 35  
 create\_unique\_dirname() (in module *optimeed.core.tools*), 31  
 createGraphs() (*LinkDataGraph* method), 29, 42, 100  
 curr\_index() (*on\_graph\_click\_showInfo.DataInformationVisuals* method), 59, 60, 78, 89, 98, 102  
 CursorMoveUp (class in *optimeed.core.ansi2html.converter*), 20  
 CYAN (*text\_format* attribute), 18, 31, 35, 42

## D

dark2() (in module *optimeed.core.color\_palette*), 24  
 DARKCYAN (*text\_format* attribute), 18, 31, 35, 42  
 Data (class in *optimeed.core*), 37  
 Data (class in *optimeed.core.graphs*), 24  
 Data (class in *optimeed.visualize*), 105  
 DataAnimationLines (class in *optimeed.visualize*), 99  
 DataAnimationLines (class in *optimeed.visualize.gui*), 91  
 DataAnimationLines (class in *optimeed.visualize.gui.widgets*), 80  
 DataAnimationLines (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick*), 63  
 DataAnimationLines (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick.on\_click\_anim*), 57  
 DataAnimationOpenGL (class in *optimeed.visualize*), 98  
 DataAnimationOpenGL (class in *optimeed.visualize.gui*), 90  
 DataAnimationOpenGL (class in *optimeed.visualize.gui.widgets*), 80  
 DataAnimationOpenGL (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick*), 62  
 DataAnimationOpenGL (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick.on\_click\_anim*), 57  
 DataAnimationOpenGLwText (class in *optimeed.visualize*), 99  
 DataAnimationOpenGLwText (class in *optimeed.visualize.gui*), 91  
 DataAnimationOpenGLwText (class in *optimeed.visualize.gui.widgets*), 80  
 DataAnimationOpenGLwText (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick*), 62  
 DataAnimationOpenGLwText (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick.on\_click\_anim*), 57  
 DataAnimationTrace (class in *optimeed.visualize.gui.gui\_data\_animation*), 82  
 DataAnimationTrace.element\_animation (class in *optimeed.visualize.gui.gui\_data\_animation*), 82  
 DataAnimationVisuals (class in *optimeed.visualize*), 94  
 DataAnimationVisuals (class in *optimeed.visualize.gui*), 85, 90  
 DataAnimationVisuals (class in *optimeed.visualize.gui.gui\_data\_animation*), 82  
 DataAnimationVisuals (class in *optimeed.visualize.gui.widgets*), 79  
 DataAnimationVisuals (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick*), 61  
 DataAnimationVisualswText (class in *optimeed.visualize*), 99  
 DataAnimationVisualswText (class in *optimeed.visualize.gui*), 91  
 DataAnimationVisualswText (class in *optimeed.visualize.gui.widgets*), 80  
 DataAnimationVisualswText (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick*), 63  
 DataAnimationVisualswText (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick.on\_click\_anim*), 57  
 DataStruct\_Interface (class in *optimeed.core*), 33  
 DataStruct\_Interface (class in *optimeed*), 33

`meed.core.collection`), 22  
`decode_str_json()` (in module `optimeed.core`), 33  
`decode_str_json()` (in module `optimeed.core.myjson`), 30  
`default` (in module `optimeed.optimize.optimizer`), 52  
`default_palette()` (in module `optimeed.core.color_palette`), 24  
`delete()` (*GraphVisual* method), 64  
`delete()` (*widget\_graphs\_visual* method), 73, 76, 86, 95, 101, 104  
`delete_all()` (*DataAnimationTrace* method), 82  
`delete_all()` (*DataAnimationVisuals* method), 62, 79, 82, 85, 90, 94  
`delete_graph()` (*widget\_graphs\_visual* method), 73, 76, 86, 95, 101, 104  
`delete_gui` (class in `optimeed.visualize.gui.widgets.graphsVisualWidget.examples.ActionsOnClick.on_click_delete`), module `optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctions_Lib`, 58  
`delete_indices_from_list()` (in module `optimeed.core`), 37  
`delete_indices_from_list()` (in module `optimeed.core.tools`), 33  
`delete_key_widgets()` (*DataAnimationLines* method), 57, 63, 80, 91, 99  
`delete_key_widgets()` (*DataAnimationOpenGL* method), 57, 62, 80, 91, 99  
`delete_lines()` (*widget\_line\_drawer* method), 62, 74, 76, 80, 87, 96  
`delete_point()` (*Data* method), 26, 39, 106  
`delete_point()` (*DataAnimationVisuals* method), 61, 79, 82, 85, 90, 94  
`delete_points_at_indices()` (*ListDataStruct* method), 23, 35  
`delete_trace()` (*GraphVisual* method), 64  
`delete_visual()` (*on\_graph\_click\_showInfo.DataInformationVisuals* method), 59, 60, 78, 89, 98, 102  
`derivate()` (in module `optimeed.core`), 37  
`derivate()` (in module `optimeed.core.tools`), 32  
`DeviceDrawerInterface` (class in `optimeed.visualize`), 97  
`DeviceDrawerInterface` (class in `optimeed.visualize.gui`), 88  
`DeviceDrawerInterface` (class in `optimeed.visualize.gui.widgets`), 81  
`DeviceDrawerInterface` (class in `optimeed.visualize.gui.widgets.openGLWidget.DeviceDrawerInterface.visualize`), module `optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctions_Lib`, 70  
`display_graphs()` (*Worker* method), 92, 103  
`DISPLAY_INFO` (*Optimizer* attribute), 53, 55  
`dist()` (in module `optimeed.core`), 36  
`dist()` (in module `optimeed.core.tools`), 32  
`DIVISION_OUTER` (*MultiObjective\_GA* attribute), 50, 54  
`do_MathsToPhys()` (*Binary\_OptimizationVariable* method), 51, 55  
`do_MathsToPhys()` (*Integer\_OptimizationVariable* method), 51, 55  
`do_MathsToPhys()` (*OptimizationVariable* method), 51  
`do_MathsToPhys()` (*Real\_OptimizationVariable* method), 51, 55  
`draw_2Dring()` (in module `optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctions_Lib`), 71  
`draw_2Dring_diff_angle()` (in module `optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctions_Lib`), 71  
`draw_carved_disk()` (in module `optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctions_Lib`), 71  
`draw_circle()` (in module `optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctions_Lib`), 71  
`draw_cylinder()` (in module `optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctions_Lib`), 71  
`draw_disk()` (in module `optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctions_Lib`), 71  
`draw_extrudeZ()` (in module `optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctions_Lib`), 71  
`draw_lines()` (in module `optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctions_Lib`), 71  
`draw_part_cylinder()` (in module `optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctions_Lib`), 71  
`draw_part_cylinder_throat()` (in module `optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctions_Lib`), 71  
`draw_part_disk()` (in module `optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctions_Lib`), 71  
`draw_part_disk_diff_angles()` (in module `optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctions_Lib`), 71  
`draw_rectangle()` (in module `optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctions_Lib`), 71  
`draw_simple_rectangle()` (in module `optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctions_Lib`), 71  
`draw_spiral()` (in module `optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctions_Lib`), 71  
`draw_spiralFront()` (in module `optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctions_Lib`), 71

*meed.visualize.gui.widgets.openGLWidget.OpenGLFunctionsLibrary*), (*widget\_graphs\_visual* method), 73, 71, 75, 86, 95, 101, 104

*draw\_spiralFull()* (in module *optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctionsLibrary*), (*MyConvergence* method), 49, 71

*draw\_spiralSheet()* (in module *optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctionsLibrary*), (*fast\_LUT\_interpolation* (class in *optimeed.core*), 37

*draw\_triList()* (in module *optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctionsLibrary*), (*fast\_LUT\_interpolation* (class in *optimeed.core.tools*), 33

*draw\_tubeSheet()* (in module *optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctionsLibrary*), (*fast\_update()* (*GraphVisual* method), 65

*drawWireTube()* (in module *optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctionsLibrary*), (*fast\_update()* (*widget\_graphs\_visual* method), 73, 71, 75, 86, 95, 101, 104

*FastObjCons* (class in *optimeed.optimize*), 54

*FastObjCons* (class in *optimeed.optimize.objAndCons*), 45

*FastObjCons* (class in *optimeed.optimize.objAndCons.fastObjCons*), 45

## E

*Emerald\_material* (in module *optimeed.visualize.gui.widgets.openGLWidget.MaterialsVisual*), 70

*encode\_str\_json()* (in module *optimeed.core*), 33

*encode\_str\_json()* (in module *optimeed.core.myjson*), 30

*END* (*text\_format* attribute), 18, 31, 35, 43

*evaluate()* (*MyProblem* method), 49

*evaluate()* (*Parametric\_analysis* method), 18, 19

*evaluate\_all()* (*MyMapEvaluator* method), 50

*evaluate\_all()* (*MyMultiprocessEvaluator* method), 50

*evaluateObjectiveAndConstraints()* (*Optimizer* method), 53, 56

*EvolutionaryConvergence* (class in *optimeed.optimize.optiAlgorithms.convergence*), 48

*EvolutionaryConvergence* (class in *optimeed.optimize.optiAlgorithms.convergence.evolutionaryConvergence*), 46

*exclude\_col()* (*HowToPlotGraph* method), 28, 41, 100

*EXCLUDED\_TAGS* (in module *optimeed.core.myjson*), 29

*export\_picture()* (*DataAnimationVisuals* method), 62, 79, 83, 86, 90, 94

*export\_str()* (*Data* method), 26, 39, 107

*export\_str()* (*Graph* method), 26, 40

*export\_str()* (*Graphs* method), 27, 41, 108

*export\_widget()* (*DataAnimationLines* method), 57, 63, 80, 91, 99

*export\_widget()* (*DataAnimationOpenGL* method), 57, 62, 80, 91, 99

*export\_xls()* (*ListDataStruct* method), 23, 35

*exportCollection()* (*gui\_collection\_exporter* method), 81, 85, 93

*figure()* (in module *optimeed.visualize*), 108

*figure\_visual()* (in module *optimeed.visualize.fastPlot*), 93

*find\_and\_replace()* (in module *optimeed.core*), 35

*find\_and\_replace()* (in module *optimeed.core.tools*), 31

*formatInfo()* (*Optimizer* method), 53, 56

*frame\_selector()* (*DataAnimationVisuals* method), 62, 79, 83, 85, 90, 94

*fromMathsToPhys()* (*MathsToPhysics* method), 44, 45, 54

*fromPhysToMaths()* (*MathsToPhysics* method), 44, 45, 54

## G

*generate()* (*MyGenerator* method), 49

*generate\_optimizationGraphs()* (*OptimizationDisplayer* method), 91, 103

*get()* (*DataAnimationTrace.element\_animation* method), 82

*get\_2D\_pareto()* (in module *optimeed.core*), 37

*get\_2D\_pareto()* (in module *optimeed.core.tools*), 32

*get\_all\_figures()* (in module *optimeed.visualize*), 109

*get\_all\_figures()* (in module *optimeed.visualize.fastPlot*), 93

*get\_all\_figures()* (*WindowHolders* method), 92, 108

*get\_all\_graphs()* (*Graphs* method), 27, 40, 107

*get\_all\_graphs\_ids()* (*Graphs* method), 27, 40, 107

*get\_all\_graphsVisual()* (*widget\_graphs\_visual* method), 73, 76, 86, 95, 101, 104

*get\_all\_id\_graphs()* (*LinkDataGraph* method), 29, 42, 100



get\_all\_options() (*Option\_class method*), 18, 30, 43  
 get\_all\_traces() (*Graph method*), 26, 40  
 get\_all\_traces() (*GraphVisual method*), 64  
 get\_all\_traces\_id\_graph() (*LinkDataGraph method*), 29, 42, 100  
 get\_analyzed\_attribute() (*Parametric\_parameter method*), 17, 19  
 get\_attr\_object() (in module *optimeed.visualize.gui.gui\_data\_selector*), 84  
 get\_attribute\_name() (*OptimizationVariable method*), 51  
 get\_attribute\_selectors() (*Container\_attribute\_selector method*), 83  
 get\_axis() (*GraphVisual method*), 64  
 get\_base\_pen() (*DataAnimationTrace method*), 82  
 get\_base\_pen() (*TraceVisual method*), 67  
 get\_base\_symbol() (*TraceVisual method*), 67  
 get\_base\_symbol\_brush() (*TraceVisual method*), 67  
 get\_base\_symbol\_pen() (*TraceVisual method*), 67  
 get\_brushes() (*TraceVisual method*), 68  
 get\_children() (*Attribute\_selector method*), 83  
 get\_children() (*Container\_attribute\_selector method*), 83  
 get\_collection() (*CollectionInfo method*), 28, 41  
 get\_collection\_from\_graph() (*LinkData-Graph method*), 29, 42, 100  
 get\_collection\_master() (*LinkData-Graph\_collection\_linker method*), 28, 41, 99  
 get\_collectionInfo() (*LinkDataGraph method*), 29, 42, 100  
 get\_color() (*Data method*), 25, 38, 105  
 get\_color() (*TraceVisual method*), 67  
 get\_colour\_background() (*DeviceDrawerInterface method*), 70, 81, 88, 97  
 get\_colour\_scalebar() (*DeviceDrawerInterface method*), 70, 81, 88, 97  
 get\_convergence() (*MultiObjective\_GA method*), 50, 51, 55  
 get\_convergence() (*NLOpt\_Algorithm method*), 49  
 get\_convergence() (*OptiHistoric method*), 52  
 get\_curr\_plotHolder() (*WindowHolders method*), 92, 108  
 get\_data() (*ListDataStruct method*), 23, 34  
 get\_data() (*TraceVisual method*), 68  
 get\_dataIndex\_from\_graphIndex() (*Data method*), 25, 38, 106  
 get\_dataIndices\_from\_graphIndices() (*Data method*), 25, 38, 106  
 get\_dataObject\_from\_graph() (*LinkData-Graph method*), 29, 42, 100  
 get\_dataObjects\_from\_graph() (*LinkData-Graph method*), 29, 42, 100  
 get\_datastruct() (*AutosaveStruct method*), 23, 34  
 get\_device() (*PipeOptimization method*), 52  
 get\_devices() (*OptiHistoric method*), 52  
 get\_element\_animations() (*DataAnimation-Trace method*), 82  
 get\_extrema\_lines() (*widget\_line\_drawer method*), 62, 74, 77, 80, 87, 96  
 get\_filename() (*AutosaveStruct method*), 22, 34  
 get\_first\_graph() (*Graphs method*), 27, 40, 107  
 get\_graph() (*Graphs method*), 27, 40, 107  
 get\_graph() (*widget\_graphs\_visual method*), 73, 76, 86, 95, 101, 104  
 get\_graph\_and\_trace\_from\_collection() (*LinkDataGraph method*), 29, 42, 100  
 get\_graphIndex\_from\_dataIndex() (*Data method*), 25, 38, 106  
 get\_graphIndices\_from\_dataIndices() (*Data method*), 25, 39, 106  
 get\_graphs() (*EvolutionaryConvergence method*), 46, 48  
 get\_graphs() (*MyConvergence method*), 49  
 get\_historic() (*PipeOptimization method*), 52  
 get\_howToPlotGraph() (*LinkDataGraph method*), 29, 42, 100  
 get\_hypervolume\_convergence() (*EvolutionaryConvergence method*), 46, 48  
 get\_id() (*CollectionInfo method*), 28, 41  
 get\_idCollection\_from\_graph() (*LinkData-Graph method*), 29, 42, 100  
 get\_indices\_points\_to\_plot() (*Data method*), 26, 39, 106  
 get\_indices\_to\_show() (*DataAnimationTrace method*), 82  
 get\_info() (*DataStruct\_Interface method*), 22, 34  
 get\_interesting\_elements() (*DataAnimation-Lines method*), 57, 63, 80, 91, 99  
 get\_interesting\_elements() (*DataAnimationOpenGLwText method*), 57, 63, 80, 91, 99  
 get\_invert\_permutations() (*Data method*), 25, 38, 105  
 get\_kwargs() (*CollectionInfo method*), 28, 41  
 get\_label\_pos() (*myAxis method*), 66  
 get\_last\_pareto() (*EvolutionaryConvergence method*), 46, 48  
 get\_layout\_buttons() (*widget\_graphs\_visual method*), 73, 76, 86, 95, 102, 104  
 get\_legend() (*Data method*), 26, 39, 106  
 get\_legend() (*GraphVisual method*), 64  
 get\_length() (*TraceVisual method*), 68  
 get\_length\_data() (*Data method*), 24, 38, 105  
 get\_linestyle() (*Data method*), 26, 39, 106  
 get\_list\_attributes() (*ListDataStruct method*),

23, 35

`get_logopti()` (*OptiHistoric method*), 52

`get_mappingData_graph()` (*LinkDataGraph method*), 29, 42, 100

`get_mappingData_trace()` (*LinkDataGraph method*), 29, 42, 100

`get_max_value()` (*Integer\_OptimizationVariable method*), 51, 55

`get_max_value()` (*Real\_OptimizationVariable method*), 51, 55

`get_min_max_attributes()` (*Attribute\_selector method*), 83

`get_min_value()` (*Integer\_OptimizationVariable method*), 51, 55

`get_min_value()` (*Real\_OptimizationVariable method*), 51, 55

`get_nadir_point()` (*EvolutionaryConvergence method*), 46, 48

`get_nadir_point_all_steps()` (*EvolutionaryConvergence method*), 46, 48

`get_name()` (*Attribute\_selector method*), 83

`get_name()` (*Container\_attribute\_selector method*), 83

`get_name()` (*FastObjCons method*), 45, 46, 54

`get_name()` (*InterfaceObjCons method*), 45, 46, 54

`get_name()` (*on\_click\_change\_symbol method*), 58, 61, 79, 90, 98

`get_name()` (*on\_click\_copy\_something method*), 58, 61, 78, 89, 98

`get_name()` (*on\_click\_extract\_pareto method*), 59, 60, 78, 89, 97

`get_name()` (*on\_graph\_click\_delete method*), 58, 60, 77, 88, 97

`get_name()` (*on\_graph\_click\_export method*), 58, 60, 78, 89, 97

`get_name()` (*on\_graph\_click\_remove\_trace method*), 59, 61, 78, 89, 98

`get_name()` (*on\_graph\_click\_showAnim method*), 57, 63, 81, 91, 99

`get_name()` (*on\_graph\_click\_showInfo method*), 59, 60, 78, 89, 98, 102

`get_name()` (*Options method*), 30, 43

`get_nb_objectives()` (*EvolutionaryConvergence method*), 46, 48

`get_nb_steps()` (*EvolutionaryConvergence method*), 46, 48

`get_ND_pareto()` (*in module optimeed.core*), 37

`get_ND_pareto()` (*in module optimeed.core.tools*), 32

`get_new_index()` (*on\_graph\_click\_showInfo.DataInformationVisual method*), 59, 60, 78, 89, 98, 102

`get_number_of_elements()` (*DataAnimation-Trace method*), 82

`get_number_of_points()` (*Data method*), 25, 38, 105

`get_object_attrs()` (*in module optimeed.core*), 36

`get_object_attrs()` (*in module optimeed.core.tools*), 31

`get_opengl_options()` (*DeviceDrawerInterface method*), 70, 81, 88, 97

`get_optionValue()` (*Option\_class method*), 18, 30, 43

`get_pareto_convergence()` (*EvolutionaryConvergence method*), 46, 48

`get_permutations()` (*Data method*), 25, 38, 105

`get_PhysToMaths()` (*Binary\_OptimizationVariable method*), 51, 55

`get_PhysToMaths()` (*Integer\_OptimizationVariable method*), 51, 55

`get_PhysToMaths()` (*OptimizationVariable method*), 51

`get_PhysToMaths()` (*Real\_OptimizationVariable method*), 51, 55

`get_plot_data()` (*Data method*), 25, 38, 105

`get_population_size()` (*EvolutionaryConvergence method*), 46, 48

`get_reference_device()` (*Parametric\_parameter method*), 17, 18

`get_results()` (*OptiHistoric method*), 52

`get_styles()` (*in module optimeed.core.ansi2html.style*), 21

`get_symbol()` (*Data method*), 26, 39, 106

`get_symbol()` (*TraceVisual method*), 67

`get_symbolOutline()` (*Data method*), 24, 38, 105

`get_symbolPens()` (*TraceVisual method*), 68

`get_symbolsSize()` (*Data method*), 24, 38, 105

`get_text_to_write()` (*ContextHandler method*), 69

`get_trace()` (*Graph method*), 26, 40

`get_trace()` (*GraphVisual method*), 64

`get_value()` (*Options method*), 30, 43

`get_values()` (*Parametric\_minmax method*), 17, 19

`get_wgGraphs()` (*in module optimeed.visualize*), 109

`get_wgGraphs()` (*in module optimeed.visualize.fastPlot*), 93

`get_wgGraphs()` (*PlotHolders method*), 92, 108

`get_wgGraphs()` (*WindowHolders method*), 92, 108

`get_widget()` (*Repr\_lines method*), 59, 61, 78, 89, 98

`get_widget()` (*Repr\_opengl method*), 60, 61, 78, 89, 98

`get_width()` (*Data method*), 25, 38, 105

`get_x()` (*Data method*), 24, 38, 105

`get_x_label()` (*Data method*), 25, 39, 106

`get_x_min()` (*Data method*), 25, 38, 105

`get_y()` (*Data method*), 25, 38, 105

`get_y_label()` (*Data method*), 26, 39, 106

`get_ylim()` (*Data method*), 25, 38, 105

`getAmb3()` (*MaterialRenderingProperties method*), 70

getDif3() (*MaterialRenderingProperties* method), 70  
 GetEar() (in module *optimeed.visualize.gui.widgets.openGLWidget.TriangulatePolygon*), 71  
 getLength() (*MultiList* method), 47  
 getLineInfo() (in module *optimeed.core*), 36  
 getLineInfo() (in module *optimeed.core.tools*), 31  
 getPath\_workspace() (in module *optimeed consolidate*), 18  
 getPath\_workspace() (in module *optimeed.core*), 33, 36  
 getPath\_workspace() (in module *optimeed.core.tools*), 31  
 getShin() (*MaterialRenderingProperties* method), 70  
 getSpec3() (*MaterialRenderingProperties* method), 70  
 Graph (class in *optimeed.core*), 39  
 Graph (class in *optimeed.core.graphs*), 26  
 graph\_clicked() (on\_click\_change\_symbol method), 57, 61, 79, 90, 98  
 graph\_clicked() (on\_click\_copy\_something method), 58, 61, 78, 89, 98  
 graph\_clicked() (on\_click\_extract\_pareto method), 59, 60, 78, 89, 97  
 graph\_clicked() (on\_graph\_click\_delete method), 58, 60, 77, 88, 97  
 graph\_clicked() (on\_graph\_click\_export method), 58, 60, 77, 88, 97  
 graph\_clicked() (on\_graph\_click\_remove\_trace method), 59, 61, 78, 89, 98  
 graph\_clicked() (on\_graph\_click\_showAnim method), 57, 63, 81, 91, 99  
 graph\_clicked() (on\_graph\_click\_showInfo method), 59, 60, 78, 89, 98, 102  
 Graphs (class in *optimeed.core*), 40  
 Graphs (class in *optimeed.core.graphs*), 26  
 Graphs (class in *optimeed.visualize*), 107  
 GraphVisual (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.graphVisual*), 63  
 GREEN (text\_format attribute), 18, 31, 35, 43  
 grid\_off() (*GraphVisual* method), 65  
 gui\_collection\_exporter (class in *optimeed.visualize*), 93  
 gui\_collection\_exporter (class in *optimeed.visualize.gui*), 84  
 gui\_collection\_exporter (class in *optimeed.visualize.gui.gui\_collection\_exporter*), 81  
 gui\_mainWindow (class in *optimeed.visualize*), 93, 100, 104  
 gui\_mainWindow (class in *optimeed.visualize.gui*), 84  
 gui\_mainWindow (class in *optimeed.visualize.gui.gui\_mainWindow*), 84  
 GuiDataSelector (class in *optimeed.visualize.gui.gui\_data\_selector*), 83  
 GuiDataPolygonGraph (class in *optimeed.visualize*), 97, 102, 108  
 guiPyqtgraph (class in *optimeed.visualize.gui*), 88  
 guiPyqtgraph (class in *optimeed.visualize.gui.widgets*), 81  
 guiPyqtgraph (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.smallGui*), 67  

## H

 hide() (*TraceVisual* method), 68  
 hide\_axes() (*GraphVisual* method), 64  
 hide\_points() (*TraceVisual* method), 67  
 HowToPlotGraph (class in *optimeed.core*), 41  
 HowToPlotGraph (class in *optimeed.core.linkDataGraph*), 28  
 HowToPlotGraph (class in *optimeed.visualize*), 100  
 hvRecursive() (*HyperVolume* method), 47  
 HyperVolume (class in *optimeed.optimize.optiAlgorithms.convergence.hypervolume*), 47  

## I

 indentParagraph() (in module *optimeed consolidate*), 18  
 indentParagraph() (in module *optimeed.core*), 33, 36  
 indentParagraph() (in module *optimeed.core.tools*), 32  
 index() (in module *optimeed.core.ansi2html.style*), 21  
 index2() (in module *optimeed.core.ansi2html.style*), 21  
 initialize() (*ContextHandler* method), 69  
 initialize() (*MyTerminationCondition* method), 49  
 initialize\_output\_collection() (*Parametric\_analysis* method), 18, 19  
 initializeOpenGL() (widget\_openGL method), 74, 77, 88, 96  
 Integer\_OptimizationVariable (class in *optimeed.optimize*), 55  
 Integer\_OptimizationVariable (class in *optimeed.optimize.optiVariable*), 51  
 integrate() (in module *optimeed.core*), 36  
 integrate() (in module *optimeed.core.tools*), 32  
 intensify() (in module *optimeed.core.ansi2html.style*), 21  
 InterfaceCharacterization (class in *optimeed.optimize*), 53  
 InterfaceCharacterization (class in *optimeed.optimize.characterization*), 44  
 InterfaceCharacterization (class in *optimeed.optimize.characterization.interfaceCharacterization*),

44  
InterfaceConvergence (class in optimeed.optimize.optiAlgorithms.convergence), 48  
InterfaceConvergence (class in optimeed.optimize.optiAlgorithms.convergence.interfaceConvergence), 48  
InterfaceDevice (class in optimeed.core), 41  
InterfaceDevice (class in optimeed.core.interfaceDevice), 28  
InterfaceMathsToPhysics (class in optimeed.optimize), 54  
InterfaceMathsToPhysics (class in optimeed.optimize.mathsToPhysics), 45  
InterfaceMathsToPhysics (class in optimeed.optimize.mathsToPhysics.interfaceMathsToPhysics), 44  
InterfaceObjCons (class in optimeed.optimize), 54  
InterfaceObjCons (class in optimeed.optimize.objAndCons), 46  
InterfaceObjCons (class in optimeed.optimize.objAndCons.interfaceObjCons), 45  
interpolate() (fast\_LUT\_interpolation method), 33, 37  
InTriangle() (in module optimeed.visualize.gui.widgets.openGLWidget.TriangulatePolygon), 71  
is\_empty() (DataAnimationVisuals method), 62, 79, 83, 85, 90, 94  
is\_monobj (EvolutionaryConvergence attribute), 46, 48  
is\_object\_selected() (in module optimeed.visualize.gui.gui\_data\_selector), 84  
is\_scattered() (Data method), 26, 39, 106  
is\_slave() (LinkDataGraph method), 29, 42, 100  
is\_slave() (LinkDataGraph.\_collection\_linker method), 28, 42, 99  
IsClockwise() (in module optimeed.visualize.gui.widgets.openGLWidget.TriangulatePolygon), 71  
IsConvex() (in module optimeed.visualize.gui.widgets.openGLWidget.TriangulatePolygon), 71  
isNonePrintMessage() (in module optimeed.core), 36  
isNonePrintMessage() (in module optimeed.core.tools), 31  
isOnWindows (in module optimeed.visualize.gui.widgets.graphsVisualWidget.pyqtgraph.Redefine), 65  
J  
json\_to\_obj() (in module optimeed.core), 33  
json\_to\_obj() (in module optimeed.core.myjson), 29  
json\_to\_obj\_safe() (in module optimeed.core), 33  
json\_to\_obj\_safe() (in module optimeed.core.myjson), 30  
K  
keyboard\_push\_action() (DeviceDrawerInterface method), 70, 81, 88, 97  
keyboardPushAction() (ContextHandler method), 69  
keyboardReleaseAction() (ContextHandler method), 69  
keyPressEvent() (gui\_mainWindow method), 84, 93, 101, 105  
keyPressEvent() (widget\_graphs\_visual method), 73, 76, 86, 95, 101, 104  
keyPressEvent() (widget\_openGL method), 74, 77, 88, 97  
KWARGS\_OPTIHISTO (Optimizer attribute), 53, 55  
L  
launch\_optimization() (OptimizationDisplayer method), 92, 103  
level() (in module optimeed.core.ansi2html.style), 21  
link\_axes() (widget\_graphs\_visual method), 73, 76, 86, 95, 101, 104  
link\_collection\_to\_graph\_collection() (LinkDataGraph method), 29, 42, 100  
LinkDataGraph (class in optimeed.core), 41  
LinkDataGraph (class in optimeed.core), 28  
LinkDataGraph (class in optimeed.visualize), 99  
LinkDataGraph.\_collection\_linker (class in optimeed.core), 41  
LinkDataGraph.\_collection\_linker (class in optimeed.core.linkDataGraph), 28  
LinkDataGraph.\_collection\_linker (class in optimeed.visualize), 99  
linkify() (in module optimeed.core.ansi2html.converter), 20  
linkXToGraph() (GraphVisual method), 64  
linspace() (in module optimeed.core), 37  
linspace() (in module optimeed.core.tools), 32  
ListDataStruct (class in optimeed.core), 34  
ListDataStruct (class in optimeed.core.collection), 23  
M  
main() (in module optimeed.core.ansi2html.converter), 21  
manage\_list() (in module optimeed.visualize.gui.gui\_data\_selector), 84  
map\_index() (DataAnimationTrace method), 82



map\_vt100\_box\_code() (in module *optimeed.core.ansi2html.converter*), 20  
 MaterialRenderingProperties (class in *optimeed.visualize.gui.widgets.openGLWidget.Materials\_visualize*), 70  
 MathsToPhysics (class in *optimeed.optimize*), 54  
 MathsToPhysics (class in *optimeed.optimize.mathsToPhysics*), 45  
 MathsToPhysics (class in *optimeed.optimize.mathsToPhysics.mathsToPhysics*), 44  
 merge() (*Graphs* method), 27, 41, 108  
 merge() (*ListDataStruct* method), 23, 35  
 meshPolygon() (in module *optimeed.visualize.gui.widgets.openGLWidget.TriangulatePolygon*), 71  
 minimumSizeHint() (*widget\_openGL* method), 74, 77, 87, 96  
 MODE\_LIGHT (in module *optimeed.visualize.gui.widgets.openGLWidget.ContextHandler*), 69  
 MODE\_ROTATION (in module *optimeed.visualize.gui.widgets.openGLWidget.ContextHandler*), 69  
 MODE\_ZOOM (in module *optimeed.visualize.gui.widgets.openGLWidget.ContextHandler*), 69  
 modify\_paintElems() (*TraceVisual.\_ModifiedPaintElem* method), 67  
 MODULE\_TAG (in module *optimeed.core.myjson*), 29  
 mouseClickAction() (*ContextHandler* method), 69  
 mouseMotionAction() (*ContextHandler* method), 69  
 mouseMoveEvent() (*widget\_openGL* method), 74, 77, 88, 97  
 mousePressEvent() (*widget\_openGL* method), 74, 77, 88, 96  
 mouseWheelAction() (*ContextHandler* method), 69  
 MultiList (class in *optimeed.optimize.optiAlgorithms.convergence.hypervolume*), 47  
 MultiList.Node (class in *optimeed.optimize.optiAlgorithms.convergence.hypervolume*), 47  
 MultiObjective\_GA (class in *optimeed.optimize*), 54  
 MultiObjective\_GA (class in *optimeed.optimize.optiAlgorithms*), 50  
 MultiObjective\_GA (class in *optimeed.optimize.optiAlgorithms.multiObjective\_GA*), 50  
 my\_fourier() (in module *optimeed.core*), 36  
 my\_fourier() (in module *optimeed.core.tools*), 32  
 myAxis (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.pyqtgraphRedefined*), 66  
 MyConvergence (class in *optimeed.optimize.optiAlgorithms.multiObjective\_GA*), 49  
 MyGenerator (class in *optimeed.optimize.optiAlgorithms.multiObjective\_GA*), 49  
 myGraphicsLayout (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.pyqtgraphRedefined*), 65  
 myGraphicsLayoutWidget (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.pyqtgraphRedefined*), 65  
 myItemSample (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.pyqtgraphRedefined*), 65  
 myLabelItem (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.pyqtgraphRedefined*), 66  
 myLabelItemHandler (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.pyqtgraphRedefined*), 66  
 myEvaluator (class in *optimeed.optimize.optiAlgorithms.multiObjective\_GA*), 50  
 myProcessEvaluator (class in *optimeed.optimize.optiAlgorithms.multiObjective\_GA*), 50  
 MyProblem (class in *optimeed.optimize.optiAlgorithms.multiObjective\_GA*), 49  
 MyTerminationCondition (class in *optimeed.optimize.optiAlgorithms.multiObjective\_GA*), 49  
 MyText (class in *optimeed.visualize.gui.widgets.openGLWidget.ContextHandler*), 69  
 myWindows (in module *optimeed.visualize*), 108  
 myWindows (in module *optimeed.visualize.fastPlot*), 93

## N

new\_figure() (*WindowHolders* method), 92, 108  
 new\_plot() (in module *optimeed.visualize*), 109  
 new\_plot() (in module *optimeed.visualize.fastPlot*), 93  
 new\_plot() (*PlotHolders* method), 92, 108  
 new\_plot() (*WindowHolders* method), 92, 108  
 next\_frame() (*DataAnimationVisuals* method), 62, 79, 83, 85, 90, 94  
 NLOpt\_Algorithm (class in *optimeed.optimize.optiAlgorithms.NLOpt\_Algorithm*), 48  
 normalize() (in module *optimeed.visualize.gui.widgets.openGLWidget.quaternions*), 66

72  
NUMBER\_OF\_CORES (MultiObjective\_GA attribute),  
50, 54  
NUMBER\_OF\_CORES (Parametric\_analysis attribute),  
17, 19  
NUMBER\_OF\_MODES (in module opti-  
meed.visualize.gui.widgets.openGLWidget.ContextHandler),  
69

## O

obj\_to\_json() (in module optimeed.core), 33  
obj\_to\_json() (in module optimeed.core.myjson), 30  
objectives (OptiHistoric.\_pointData attribute), 52  
objectives\_per\_step (EvolutionaryConvergence  
attribute), 46, 48  
on\_click() (widget\_graphs\_visual method), 72, 75,  
86, 95, 101, 103  
on\_click\_change\_symbol (class in opti-  
meed.visualize), 98  
on\_click\_change\_symbol (class in opti-  
meed.visualize.gui), 89  
on\_click\_change\_symbol (class in opti-  
meed.visualize.gui.widgets), 78  
on\_click\_change\_symbol (class in opti-  
meed.visualize.gui.widgets.graphsVisualWidget.examplesActionO  
nClick), 61  
on\_click\_change\_symbol (class in opti-  
meed.visualize.gui.widgets.graphsVisualWidget.examplesActionO  
nClick), 57  
on\_click\_copy\_something (class in opti-  
meed.visualize), 98  
on\_click\_copy\_something (class in opti-  
meed.visualize.gui), 89  
on\_click\_copy\_something (class in opti-  
meed.visualize.gui.widgets), 78  
on\_click\_copy\_something (class in opti-  
meed.visualize.gui.widgets.graphsVisualWidget.examplesActionO  
nClick), 61  
on\_click\_copy\_something (class in opti-  
meed.visualize.gui.widgets.graphsVisualWidget.examplesActionO  
nClick), 58  
on\_click\_extract\_pareto (class in opti-  
meed.visualize), 97  
on\_click\_extract\_pareto (class in opti-  
meed.visualize.gui), 89  
on\_click\_extract\_pareto (class in opti-  
meed.visualize.gui.widgets), 78  
on\_click\_extract\_pareto (class in opti-  
meed.visualize.gui.widgets.graphsVisualWidget.examplesActionO  
nClick), 60  
on\_click\_extract\_pareto (class in opti-  
meed.visualize.gui.widgets.graphsVisualWidget.examplesActionO  
nClick), 59

on\_graph\_click\_delete (class in opti-  
meed.visualize), 97  
on\_graph\_click\_delete (class in opti-  
meed.visualize.gui), 88  
on\_graph\_click\_delete (class in opti-  
meed.visualize.gui.widgets), 77  
on\_graph\_click\_delete (class in opti-  
meed.visualize.gui.widgets.graphsVisualWidget.examplesActionO  
nClick), 60  
on\_graph\_click\_delete (class in opti-  
meed.visualize.gui.widgets.graphsVisualWidget.examplesActionO  
nClick), 58  
on\_graph\_click\_export (class in opti-  
meed.visualize), 97  
on\_graph\_click\_export (class in opti-  
meed.visualize.gui), 88  
on\_graph\_click\_export (class in opti-  
meed.visualize.gui.widgets), 77  
on\_graph\_click\_export (class in opti-  
meed.visualize.gui.widgets.graphsVisualWidget.examplesActionO  
nClick), 60  
on\_graph\_click\_export (class in opti-  
meed.visualize.gui.widgets.graphsVisualWidget.examplesActionO  
nClick), 58  
on\_graph\_click\_interface (class in opti-  
meed.visualize), 98  
on\_graph\_click\_interface (class in opti-  
meed.visualize.gui), 90  
on\_graph\_click\_interface (class in opti-  
meed.visualize.gui.widgets), 79  
on\_graph\_click\_interface (class in opti-  
meed.visualize.gui.widgets.graphsVisualWidget.examplesActionO  
nClick), 61  
on\_graph\_click\_interface (class in opti-  
meed.visualize.gui.widgets.widget\_graphs\_visual),  
72  
on\_graph\_click\_remove\_trace (class in opti-  
meed.visualize), 98  
on\_graph\_click\_remove\_trace (class in opti-  
meed.visualize.gui), 89  
on\_graph\_click\_remove\_trace (class in opti-  
meed.visualize.gui.widgets), 78  
on\_graph\_click\_remove\_trace (class in opti-  
meed.visualize.gui.widgets.graphsVisualWidget.examplesActionO  
nClick), 61  
on\_graph\_click\_remove\_trace (class in opti-  
meed.visualize.gui.widgets.graphsVisualWidget.examplesActionO  
nClick), 59  
on\_graph\_click\_showAnim (class in opti-  
meed.visualize), 99  
on\_graph\_click\_showAnim (class in opti-  
meed.visualize.gui), 91  
on\_graph\_click\_showAnim (class in opti-  
meed.visualize.gui.widgets), 81

on\_graph\_click\_showAnim (class in optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick), 63  
 on\_graph\_click\_showAnim (class in optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick), 57  
 on\_graph\_click\_showInfo (class in optimeed.visualize), 97, 102  
 on\_graph\_click\_showInfo (class in optimeed.visualize.gui), 89  
 on\_graph\_click\_showInfo (class in optimeed.visualize.gui.widgets), 78  
 on\_graph\_click\_showInfo (class in optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick), 60  
 on\_graph\_click\_showInfo (class in optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick), 59  
 on\_graph\_click\_showInfo.DataInformationVisuals (class in optimeed.visualize), 98, 102  
 on\_graph\_click\_showInfo.DataInformationVisuals (module), 45  
 on\_graph\_click\_showInfo.DataInformationVisuals (class in optimeed.visualize.gui), 89  
 on\_graph\_click\_showInfo.DataInformationVisuals (module), 45  
 on\_graph\_click\_showInfo.DataInformationVisuals (class in optimeed.visualize.gui.widgets), 78  
 on\_graph\_click\_showInfo.DataInformationVisuals (class in optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick), 60  
 on\_graph\_click\_showInfo.DataInformationVisuals (class in optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick), 59  
 on\_update\_signal() (widget\_line\_drawer method), 62, 74, 76, 80, 87, 96  
 OPTI\_ALGORITHM (MultiObjective\_GA attribute), 50, 54  
 OptiHistoric (class in optimeed.optimize.optimizer), 52  
 OptiHistoric.\_pointData (class in optimeed.optimize.optimizer), 52  
 optimeed (module), 17  
 optimeed consolidate (module), 17  
 optimeed consolidate.parametric\_analysis (module), 17  
 optimeed.core (module), 19  
 optimeed.core ansi2html (module), 19  
 optimeed.core ansi2html.converter (module), 19  
 optimeed.core ansi2html.style (module), 21  
 optimeed.core ansi2html.util (module), 21  
 optimeed.core.collection (module), 22  
 optimeed.core.color\_palette (module), 24  
 optimeed.core.commonImport (module), 24  
 optimeed.core.graphs (module), 24  
 optimeed.core.interfaceDevice (module), 28  
 optimeed.core.linkDataGraph (module), 28  
 optimeed.core.myjson (module), 29  
 optimeed.core.options (module), 30  
 optimeed.core.options.click (module), 30  
 optimeed.optimize (module), 43  
 optimeed.optimize.characterization (module), 43  
 optimeed.optimize.characterization.characterization (module), 43  
 optimeed.optimize.characterization.interfaceCharacterization (module), 44  
 optimeed.optimize.mathsToPhysics (module), 44  
 optimeed.optimize.mathsToPhysics.interfaceMathsToPhysics (module), 44  
 optimeed.optimize.mathsToPhysics.mathsToPhysics (module), 44  
 optimeed.optimize.objAndCons (module), 45  
 optimeed.optimize.objAndCons.fastObjCons (module), 45  
 optimeed.optimize.objAndCons.interfaceObjCons (module), 45  
 optimeed.optimize.optiAlgorithms (module), 46  
 optimeed.optimize.optiAlgorithms.algorithmInterface (module), 46  
 optimeed.optimize.optiAlgorithms.convergence (module), 46  
 optimeed.optimize.optiAlgorithms.convergence.evolution (module), 46  
 optimeed.optimize.optiAlgorithms.convergence.hyper (module), 47  
 optimeed.optimize.optiAlgorithms.convergence.inter (module), 48  
 optimeed.optimize.optiAlgorithms.multiObjective\_GA (module), 49  
 optimeed.optimize.optiAlgorithms.NLOpt\_Algorithm (module), 48  
 optimeed.optimize.optimizer (module), 52  
 optimeed.optimize.optiVariable (module), 51  
 optimeed.visualize (module), 56  
 optimeed.visualize.displayOptimization (module), 91  
 optimeed.visualize.fastPlot (module), 92  
 optimeed.visualize.gui (module), 56  
 optimeed.visualize.gui.gui\_collection\_exporter (module), 81  
 optimeed.visualize.gui.gui\_data\_animation (module), 81  
 optimeed.visualize.gui.gui\_data\_selector (module), 83  
 optimeed.visualize.gui.gui\_mainWindow





prepare() (*Ansi2HTMLConverter method*), 21, 22  
 preProcess() (*HyperVolume method*), 47  
 printIfShown() (*in module optimeed.core*), 33, 36, 37  
 printIfShown() (*in module optimeed.core.tools*), 31  
 produce\_headers() (*Ansi2HTMLConverter method*), 21, 22  
 PURPLE (*text\_format attribute*), 18, 31, 35, 42

## Q

q\_conjugate() (*in module optimeed.visualize.gui.widgets.openGLWidget.quaternions*), 72  
 q\_mult() (*in module optimeed.visualize.gui.widgets.openGLWidget.quaternions*), 72  
 q\_to\_axisangle() (*in module optimeed.visualize.gui.widgets.openGLWidget.quaternions*), 72  
 q\_to\_mat4() (*in module optimeed.visualize.gui.widgets.openGLWidget.quaternions*), 72  
 quicksort() (*in module optimeed.core*), 36  
 quicksort() (*in module optimeed.core.tools*), 31  
 qv\_mult() (*in module optimeed.visualize.gui.widgets.openGLWidget.quaternions*), 72

## R

read\_to\_unicode() (*in module optimeed.core.ansi2html.util*), 21  
 Real\_OptimizationVariable (*class in optimeed.optimize*), 55  
 Real\_OptimizationVariable (*class in optimeed.optimize.optiVariable*), 51  
 RED (*text\_format attribute*), 18, 31, 35, 43  
 Red\_material (*in module optimeed.visualize.gui.widgets.openGLWidget.Materials\_visual*), 70  
 redraw() (*ContextHandler method*), 69  
 reformatXYtoList() (*in module optimeed.visualize.gui.widgets.openGLWidget.TriangulatePolygon*), 71  
 refreshTraceList() (*guiPyqtgraph method*), 67, 81, 88, 97, 102, 108  
 reinsert() (*MultiList method*), 47  
 remove() (*MultiList method*), 47  
 remove\_element\_from\_graph() (*LinkData-Graph method*), 29, 42, 100  
 remove\_elements\_from\_trace() (*LinkData-Graph method*), 29, 42, 100  
 remove\_feature() (*GraphVisual method*), 64  
 remove\_graph() (*Graphs method*), 27, 41, 107  
 remove\_trace() (*Graph method*), 26, 39

remove\_trace() (*Graphs method*), 27, 40, 107  
 remove\_trace() (*LinkDataGraph method*), 29, 42, 100

Repr\_lines (*class in optimeed.visualize*), 98  
 Repr\_lines (*class in optimeed.visualize.gui*), 89  
 Repr\_lines (*class in optimeed.visualize.gui.widgets*), 78

Repr\_lines (*class in optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionO*), 61

Repr\_lines (*class in optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionO*), 59

Repr\_opengl (*class in optimeed.visualize*), 98  
 Repr\_opengl (*class in optimeed.visualize.gui*), 89

Repr\_opengl (*class in optimeed.visualize.gui.widgets*), 78

Repr\_opengl (*class in optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionO*), 60

Repr\_opengl (*class in optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionO*), 59

reset() (*\_State method*), 20

reset() (*AlgorithmInterface method*), 49

reset() (*Graphs method*), 27, 41, 108

reset() (*gui\_collection\_exporter method*), 81, 85, 93

reset() (*on\_graph\_click\_delete method*), 58, 60, 77, 88, 97

reset() (*PlotHolders method*), 92, 108

reset() (*TraceVisual.\_ModifiedPaintElem method*), 67

reset\_all() (*DataAnimationVisuals method*), 61, 79, 82, 85, 90, 94

reset\_all\_brushes() (*TraceVisual method*), 68

reset\_all\_symbolPens() (*TraceVisual method*), 69

reset\_brush() (*TraceVisual method*), 68

reset\_graph() (*on\_graph\_click\_export method*), 58, 60, 78, 88, 97

reset\_paintElem() (*TraceVisual.\_ModifiedPaintElem method*), 67

reset\_symbol() (*TraceVisual method*), 68

reset\_symbolPen() (*TraceVisual method*), 69

resizeEvent() (*myAxis method*), 66

resizeGL() (*widget\_openGL method*), 74, 77, 88, 96

resizeWindowAction() (*ContextHandler method*), 69

rgetattr() (*in module optimeed consolidate*), 18

rgetattr() (*in module optimeed.core*), 33, 36

rgetattr() (*in module optimeed.core.tools*), 32

rsetattr() (*in module optimeed consolidate*), 18

rsetattr() (*in module optimeed.core*), 36

rsetattr() (*in module optimeed.core.tools*), 31

Rule (*class in optimeed.core.ansi2html.style*), 21

run() (*DataAnimationVisuals* method), 62, 79, 83, 85, 90, 94  
 run() (*gui\_mainWindow* method), 84, 93, 101, 104  
 run() (*GuiDataSelector* method), 83  
 run() (*Parametric\_analysis* method), 17, 19  
 run\_optimization() (*Optimizer* method), 53, 56

## S

save() (*AutosaveStruct* method), 23, 34  
 save() (*ListDataStruct* method), 23, 34  
 save() (*OptiHistoric* method), 52  
 SCHEME (in module *optimeed.core.ansi2html.style*), 21  
 scrollable\_widget\_text (class in *optimeed.visualize.gui.widgets.widget\_text*), 75  
 set\_actionOnClick() (*widget\_graphs\_visual* method), 73, 76, 87, 95, 102, 104  
 set\_actionOnClose() (*gui\_mainWindow* method), 84, 93, 100, 104  
 set\_actionsOnClick() (*OptimizationDisplayer* method), 91, 103  
 set\_all\_options() (*Option\_class* method), 18, 30, 43  
 set\_article\_template() (*widget\_graphs\_visual* method), 73, 76, 87, 96, 102, 104  
 set\_attribute\_data() (*ListDataStruct* method), 23, 34  
 set\_attribute\_equation() (*ListDataStruct* method), 23, 34  
 set\_attribute\_selectors() (*Container\_attribute\_selector* method), 83  
 set\_brush() (*TraceVisual* method), 68  
 set\_brushes() (*TraceVisual* method), 68  
 set\_collection() (*gui\_collection\_exporter* method), 81, 85, 94  
 set\_color() (*TraceVisual* method), 67  
 set\_color\_palette() (*GraphVisual* method), 64  
 set\_convergence() (*OptiHistoric* method), 52  
 set\_curr\_brush() (*DataAnimationTrace* method), 82  
 set\_currFigure() (*WindowHolders* method), 92, 108  
 set\_data() (*Data* method), 24, 37, 105  
 set\_data() (*ListDataStruct* method), 23, 34  
 set\_data\_at\_index() (*ListDataStruct* method), 23, 34  
 set\_device() (*PipeOptimization* method), 52  
 set\_deviceDrawer() (*ContextHandler* method), 69  
 set\_deviceDrawer() (*widget\_openGL* method), 74, 77, 87, 96  
 set\_deviceToDraw() (*ContextHandler* method), 69  
 set\_deviceToDraw() (*widget\_openGL* method), 74, 77, 88, 96

set\_evaluationFunction() (*MultiObjective\_GA* method), 50, 54  
 set\_evaluationFunction() (*NLOpt\_Algorithm* method), 49  
 set\_filename() (*AutosaveStruct* method), 22, 34  
 set\_font() (*myLegend* method), 66  
 set\_fontLabel() (*GraphVisual* method), 63  
 set\_fontLegend() (*GraphVisual* method), 64  
 set\_fontTicks() (*GraphVisual* method), 63  
 set\_graph\_disposition() (*myGraphicsLayout* method), 65  
 set\_graph\_disposition() (*widget\_graphs\_visual* method), 72, 75, 86, 95, 101, 103  
 set\_graph\_properties() (*GraphVisual* method), 64  
 set\_historic() (*PipeOptimization* method), 52  
 set\_idle\_brush() (*DataAnimationTrace* method), 82  
 set\_indices\_points\_to\_plot() (*Data* method), 26, 39, 106  
 set\_info() (*DataStruct\_Interface* method), 22, 34  
 set\_info() (*gui\_collection\_exporter* method), 81, 85, 94  
 set\_info() (*OptiHistoric* method), 52  
 set\_label\_pos() (*GraphVisual* method), 64  
 set\_label\_pos() (*myAxis* method), 66  
 set\_legend() (*GraphVisual* method), 64  
 set\_lims() (*GraphVisual* method), 64  
 set\_lines() (*widget\_line\_drawer* method), 62, 74, 76, 80, 87, 96  
 set\_max\_opti\_time() (*Optimizer* method), 53, 56  
 set\_maxtime() (*MultiObjective\_GA* method), 50, 54  
 set\_maxtime() (*NLOpt\_Algorithm* method), 49  
 set\_number\_ticks() (*myAxis* method), 66  
 set\_numberTicks() (*GraphVisual* method), 63  
 set\_offset() (*myItemSample* method), 65  
 set\_offset\_sample() (*myLegend* method), 66  
 set\_optimizer() (*Optimizer* method), 53, 55  
 set\_option() (*Options* method), 30, 43  
 set\_optionValue() (*Option\_class* method), 18, 30, 43  
 set\_permutations() (*Data* method), 25, 39, 106  
 set\_points\_at\_step() (*EvolutionaryConvergence* method), 46, 48  
 set\_pop\_size() (*ConvergenceManager* method), 48  
 set\_position() (*myLegend* method), 66  
 set\_refreshTime() (*DataAnimationVisuals* method), 62, 79, 83, 85, 90, 94  
 set\_results() (*OptiHistoric* method), 52  
 set\_same\_master() (*LinkData-Graph\_collection\_linker* method), 28, 42, 99  
 set\_self() (*Options* method), 30, 43

- set\_space\_sample\_label() (*myLegend method*), 66  
 set\_specialButtonsMapping() (*ContextHandler method*), 69  
 set\_symbol() (*TraceVisual method*), 68  
 set\_symbolPen() (*TraceVisual method*), 68  
 set\_symbolPens() (*TraceVisual method*), 69  
 set\_text() (*scrollable\_widget\_text method*), 75  
 set\_text() (*widget\_text method*), 62, 75, 77, 80, 88, 97  
 set\_title() (*GraphVisual method*), 64  
 set\_title() (*in module optimeed.visualize*), 109  
 set\_title() (*in module optimeed.visualize.fastPlot*), 93  
 set\_title() (*PlotHolders method*), 92, 108  
 set\_title() (*widget\_graphs\_visual method*), 73, 76, 87, 95, 102, 104  
 set\_title() (*WindowHolders method*), 92, 108  
 set\_width\_cell() (*myItemSample method*), 65  
 set\_width\_cell\_sample() (*myLegend method*), 66  
 setText() (*myLabelItem method*), 66  
 shouldTerminate() (*MyTerminationCondition method*), 50  
 show() (*in module optimeed.visualize*), 108  
 show() (*in module optimeed.visualize.fastPlot*), 93  
 show() (*TraceVisual method*), 68  
 show() (*WindowHolders method*), 92, 108  
 show\_all() (*DataAnimationTrace method*), 82  
 show\_all() (*DataAnimationVisuals method*), 62, 79, 82, 85, 90, 94  
 SHOW\_CURRENT (*in module optimeed.core*), 37, 41  
 SHOW\_CURRENT (*in module optimeed.core.commonImport*), 24  
 SHOW\_DEBUG (*in module optimeed.core*), 37, 41  
 SHOW\_DEBUG (*in module optimeed.core.commonImport*), 24  
 SHOW\_ERROR (*in module optimeed.core*), 37, 41  
 SHOW\_ERROR (*in module optimeed.core.commonImport*), 24  
 SHOW\_INFO (*in module optimeed.core*), 37, 41  
 SHOW\_INFO (*in module optimeed.core.commonImport*), 24  
 SHOW\_WARNING (*in module optimeed.core*), 33, 37, 41  
 SHOW\_WARNING (*in module optimeed.core.commonImport*), 24  
 showEvent() (*widget\_menuButton method*), 74, 77, 87, 96  
 signal\_graph\_changed (*widget\_graphs\_visual attribute*), 72, 75, 86, 94, 101, 103  
 signal\_has\_exported (*gui\_collection\_exporter attribute*), 81, 85, 93  
 signal\_has\_reset (*gui\_collection\_exporter attribute*), 81, 85, 93  
 signal\_must\_update (*TraceVisual attribute*), 67  
 signal\_must\_update (*widget\_graphs\_visual attribute*), 72, 75, 86, 94, 101, 103  
 signal\_must\_update (*widget\_line\_drawer attribute*), 62, 74, 76, 80, 87, 96  
 signal\_optimization\_over (*OptimizationDisplay attribute*), 91, 102  
 signal\_show\_UI (*Worker attribute*), 92, 103  
 Silver\_material (*in module optimeed.visualize.gui.widgets.openGLWidget.Materials\_visual*), 70  
 sizeHint() (*widget\_openGL method*), 74, 77, 87, 96  
 slider\_handler() (*DataAnimationVisuals method*), 62, 79, 83, 85, 90, 94  
 SLIDER\_MAXIMUM\_VALUE (*DataAnimationVisuals attribute*), 61, 79, 82, 85, 90, 94  
 SLIDER\_MINIMUM\_VALUE (*DataAnimationVisuals attribute*), 61, 79, 82, 85, 90, 94  
 software\_version() (*in module optimeed.core*), 35  
 software\_version() (*in module optimeed.core.tools*), 31  
 sortByDimension() (*HyperVolume method*), 47  
 sparse\_subset() (*in module optimeed.core*), 36  
 sparse\_subset() (*in module optimeed.core.tools*), 32  
 SpecialButtonsMapping (*class in optimeed.visualize.gui.widgets.openGLWidget.ContextHandler*), 69  
 start\_autosave() (*AutosaveStruct method*), 22, 34  
 start\_qt\_mainloop() (*in module optimeed.visualize*), 93, 105  
 start\_qt\_mainloop() (*in module optimeed.visualize.gui*), 84  
 start\_qt\_mainloop() (*in module optimeed.visualize.gui.gui\_mainWindow*), 84  
 Steel\_material (*in module optimeed.visualize.gui.widgets.openGLWidget.Materials\_visual*), 70  
 stop\_autosave() (*AutosaveStruct method*), 22, 34  
 stop\_qt\_mainloop() (*in module optimeed.visualize*), 93, 105  
 stop\_qt\_mainloop() (*in module optimeed.visualize.gui*), 84  
 stop\_qt\_mainloop() (*in module optimeed.visualize.gui.gui\_mainWindow*), 84  
 str\_all\_attr() (*in module optimeed.core*), 37  
 str\_all\_attr() (*in module optimeed.core.tools*), 32  
 symbol\_isfilled() (*Data method*), 24, 38, 105
- ## T
- text\_format (*class in optimeed consolidate*), 18  
 text\_format (*class in optimeed.core*), 35, 42  
 text\_format (*class in optimeed.core.tools*), 31  
 theActionOnUpdate (*GuiDataSelector attribute*), 83

time (*OptiHistoric.\_pointData* attribute), 52  
 to\_css\_classes() (*\_State* method), 20  
 toggle() (*TraceVisual* method), 68

TraceVisual (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.traceVisual*), 67

TraceVisual.\_ModifiedPaintElem (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.traceVisual*), 67

truncate() (in module *optimeed.core*), 37  
 truncate() (in module *optimeed.core.tools*), 32

## U

UNDERLINE (*text\_format* attribute), 18, 31, 35, 43  
 universalPath() (in module *optimeed.core*), 36  
 universalPath() (in module *optimeed.core.tools*), 31  
 update() (*GraphVisual* method), 65  
 update\_graphs() (*LinkDataGraph* method), 29, 42, 100  
 update\_graphs() (*widget\_graphs\_visual* method), 72, 75, 86, 95, 101, 103  
 update\_widget\_w\_animation() (*DataAnimationLines* method), 57, 63, 80, 91, 99  
 update\_widget\_w\_animation() (*DataAnimationOpenGL* method), 57, 62, 80, 91, 99  
 update\_widget\_w\_animation() (*DataAnimationOpenGLwText* method), 57, 63, 80, 91, 99  
 update\_widget\_w\_animation() (*DataAnimationVisualswText* method), 57, 63, 80, 91, 99  
 updateChildren() (*Graphs* method), 26, 40, 107  
 updateSize() (*myLegend* method), 66  
 updateTrace() (*TraceVisual* method), 67  
 useOpenGL() (*myGraphicsLayoutWidget* method), 65

## V

value (*Bar* attribute), 30  
 VERSION (in module *optimeed*), 109  
 VT100\_BOX\_CODES (in module *optimeed.core.ansi2html.converter*), 20

## W

wheelEvent() (*widget\_openGL* method), 74, 77, 88, 97  
 WHITE (*text\_format* attribute), 18, 31, 35, 43  
 widget\_graphs\_visual (class in *optimeed.visualize*), 94, 101, 103  
 widget\_graphs\_visual (class in *optimeed.visualize.gui*), 86  
 widget\_graphs\_visual (class in *optimeed.visualize.gui.widgets*), 75

widget\_graphs\_visual (class in *optimeed.visualize.gui.widgets.widget\_graphs\_visual*), 72

widget\_line\_drawer (class in *optimeed.visualize*), 96

widget\_line\_drawer (class in *optimeed.visualize.gui*), 87

widget\_line\_drawer (class in *optimeed.visualize.gui.widgets*), 76, 80

widget\_line\_drawer (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionO*), 62

widget\_line\_drawer (class in *optimeed.visualize.gui.widgets.widget\_line\_drawer*), 73

widget\_menuButton (class in *optimeed.visualize*), 96

widget\_menuButton (class in *optimeed.visualize.gui*), 87

widget\_menuButton (class in *optimeed.visualize.gui.widgets*), 77

widget\_menuButton (class in *optimeed.visualize.gui.widgets.widget\_menuButton*), 74

widget\_openGL (class in *optimeed.visualize*), 96

widget\_openGL (class in *optimeed.visualize.gui*), 87

widget\_openGL (class in *optimeed.visualize.gui.widgets*), 77

widget\_openGL (class in *optimeed.visualize.gui.widgets.widget\_openGL*), 74

widget\_text (class in *optimeed.visualize*), 97

widget\_text (class in *optimeed.visualize.gui*), 88

widget\_text (class in *optimeed.visualize.gui.widgets*), 77, 79

widget\_text (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionO*), 62

widget\_text (class in *optimeed.visualize.gui.widgets.widget\_text*), 75

WindowHolders (class in *optimeed.visualize*), 108

WindowHolders (class in *optimeed.visualize.fastPlot*), 92

Worker (class in *optimeed.visualize*), 103

Worker (class in *optimeed.visualize.displayOptimization*), 92

## Y

YELLOW (*text\_format* attribute), 18, 31, 35, 43

Yellow\_Emerald\_material (in module *optimeed.visualize.gui.widgets.openGLWidget.Materials\_visual*), 70