

---

**optimeed**

***Release 1.1.2***

**Mar 12, 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
3.3	Loading and saving data . . . . .	12
<b>4</b>	<b>Gallery</b>	<b>15</b>
4.1	Gallery . . . . .	15
<b>5</b>	<b>License and support</b>	<b>17</b>
5.1	License and Support . . . . .	17
<b>6</b>	<b>API</b>	<b>19</b>
6.1	<b>optimeed</b> . . . . .	19
<b>7</b>	<b>Developer guide</b>	<b>115</b>
7.1	Developer documentation . . . . .	115
	<b>Python Module Index</b>	<b>117</b>
	<b>Index</b>	<b>119</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.

```
"""This example shows how to start a small optimization problem. Start with these_
↳ imports: (note: full path is not necessary)"""

from optimeed.core import InterfaceDevice
from optimeed.optimize.optiAlgorithms import MultiObjective_GA as_
↳ OptimizationAlgorithm
```

(continues on next page)

(continued from previous page)

```

# from optimeed.optimize.optiAlgorithms import NLOpt_Algorithm as_
↳ OptimizationAlgorithm # Toggle this line to use NLOpt
from optimeed.optimize import Optimizer, Real_OptimizationVariable, InterfaceObjCons,
↳ InterfaceCharacterization
from optimeed.visualize.displayOptimization import OptimizationDisplayer
from optimeed.visualize import start_qt_mainloop
import time

"""User-defined structures"""

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

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

class Characterization(InterfaceCharacterization):
    """Define the Characterization scheme. In this case nothing is performed,
    but this is typically where model code will be executed and results saved."""
    def compute(self, thedevice):
        time.sleep(0.005)

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

class MyObjective2(InterfaceObjCons):
    """Second objective function (to be minimized)"""
    def compute(self, theDevice):
        return 4 + 2*(theDevice.y+3)**0.5*(1+(theDevice.x-1)**2)**0.5

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

def run():
    """Start the main code. Instantiate previously defined classes."""
    theDevice = Device()
    theAlgo = OptimizationAlgorithm()
    theAlgo.set_optionValue(theAlgo.NUMBER_OF_CORES, 2) # Toggle this line to use_
↳ more cores. Default is 1 (single core)

    theCharacterization = Characterization()

    """Variable to be optimized"""

```

(continues on next page)

(continued from previous page)

```

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

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

"""Set the optimizer"""
theOptimizer = Optimizer()
theOptimizer.set_optionValue(theOptimizer.KWARGS_OPTIHISTO, {"autosave": True})
PipeOptimization = theOptimizer.set_optimizer(theDevice, listOfObjectives,
↳listOfConstraints, optimizationVariables, theOptimizationAlgorithm=theAlgo,
↳theCharacterization=theCharacterization)
theOptimizer.set_max_opti_time(3)

"""Start the optimization"""
display_opti = True
if display_opti: # Display real-time graphs
    optiDisplayer = OptimizationDisplayer(PipeOptimization, listOfObjectives,
↳theOptimizer)
    _, _, _ = optiDisplayer.generate_optimizationGraphs()
    resultsOpti, convergence = optiDisplayer.launch_optimization()
else: # Just focus on results
    resultsOpti, convergence = theOptimizer.run_optimization()

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

if display_opti:
    start_qt_mainloop() # To keep windows alive

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

```

## 3.2 Quickstart Visualization

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

- A QMainWindow that includes layouts, (ex: horizontal, vertical, grid, ...)

- Layouts can include widgets.
- Widgets can be anything: buttons, menu, opengl 3D representation, graphs, ... Several high-level widgets are proposed, check `optimeed.visualize.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 *

class Cone(InterfaceDevice):
    """Device to be drawn"""
    def __init__(self):
        self.width = 1 # base width
        self.height = 1.5 # height

class ConeDrawer(DeviceDrawerInterface):
    """Drawer of the device"""
    def __init__(self):
        self.theCone = None

    def draw(self, theCone):
        self.theCone = theCone
        glPushMatrix() # Remove the previous matrices transformations
        glTranslate(0, 0, -theCone.height/2) # Move the cone
        Bronze_material.activateMaterialProperties() # Change colour aspect of the ↪
↪cones
        draw_disk(0, theCone.width, 50, translate=theCone.height) # Draw the base
        gluCylinder(gluNewQuadric(), 0, theCone.width, theCone.height, 50, 10) # ↪
↪Draw the cylinde
        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.x += 0.2 # Change the radius length when h is pressed

def run():
    """Instantiates objects and run the code"""
    openGLWidget = widget_openGL()
    theDrawer = ConeDrawer()

```

(continues on next page)

(continued from previous page)

```

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

```

### 3.2.2 Advanced visualization:

```

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

from optimeed.core import ListDataStruct
# 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

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

def run():
    """Example on how to get back data from optimization"""

    """Load collections. File is relative to this directory __file__"""
    foldername = os.path.join(os.path.dirname(__file__), 'resources')
    collection_devices = ListDataStruct.load(foldername + '/autosaved.json')
    collection_logOpti = ListDataStruct.load(foldername + '/logopti.json')

    """Instantiates high level module that links the data contained in collections to_
↪graphs (that will be later created)"""
    theDataLink = LinkDataGraph()

```

(continues on next page)

(continued from previous page)

```

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])
myWindow.run(True)

```

### 3.3 Loading and saving data

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

Imagine the following structure to be saved:

```

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

```

(continues on next page)



(continued from previous page)

```

        self.R_out = 5e-3

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

```

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

```

class TopoA:
    R_in: float
    R_out: float

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

class MyMotor:
    rotor: TopoA
    length: float

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

```

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

```

from optimeed.core import SaveableObject

class TopoA:
    R_in: float
    R_out: float

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

class MyMotor(SaveableObject):
    length: float

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

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

```

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

using “`json.load`”. Then, convert the dictionary by using `json_to_obj()`

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

## CHAPTER 4

---

### Gallery

---

#### 4.1 Gallery



### 5.1 License and Support

#### 5.1.1 License

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

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

#### 5.1.2 Support

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



## 6.1 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, minValue, maxValue, 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*)

## additional\_tools

## Module Contents

```
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

**interpolate\_table** (*x0, x\_values, y\_values*)

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

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

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

**reconstitute\_signal** (*amplitudes, phases, numberOfPeriods=1, x\_points=None, n\_points=50*)

Reconstitute the signal from fft. Number of periods of the signal must be specified if different of 1

**my\_fft** (*y*)

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

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

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

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

**quicksort** (*array*)

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

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

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

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

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

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

#### Parameters

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

**Returns** integral value

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

Fourier analys

#### Parameters

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

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

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

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

**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'

**\_\_getstate\_\_** (*self*)**\_\_setstate\_\_** (*self*, *state*)**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

**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 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*)

## 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*, *device\_to\_assign*, *resetAttribute=False*)

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

**Parameters**

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

## linkDataGraph

### Module Contents

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

**class SaveableObject**

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

**`__get_object_class`** (*theObj*)

**`__get_object_module`** (*theObj*)

**`__object_to_FQCN`** (*theobj*)

Gets module path of object

**`__find_class`** (*moduleName, className*)

**`json_to_obj`** (*json\_dict*)

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

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

**Raises**

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

**`json_to_obj_safe`** (*json\_dict, cls*)

Safe class to create object from dictionary.

**Parameters**

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

**`__instantiates_annotated_object`** (*\_json\_dict, \_cls*)

**`__get_annotations`** (*theObj*)

**`obj_to_json`** (*theObj*)

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

**`encode_str_json`** (*theStr*)

**`decode_str_json`** (*theStr*)

## 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*)

**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*)

**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.

**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. Inheritance of annotation is managed by this function

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

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

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

**Raises**

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

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

Safe class to create object from dictionary.

**Parameters**

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

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

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

**class SaveableObject**

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

**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`

    __getstate__ (self)

    __setstate__ (self, state)

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

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

**blackOnly** (*N*)

**dark2** (*N*)

**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*)

**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*)

**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.

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

**Parameters**

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

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

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

```

**interpolate\_table** (*x0, x\_values, y\_values*)

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

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

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

**reconstitute\_signal** (*amplitudes, phases, numberOfPeriods=1, x\_points=None, n\_points=50*)

Reconstitute the signal from fft. Number of periods of the signal must be specified if different of 1

**my\_fft** (*y*)

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

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

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

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

**quicksort** (*array*)

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

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

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

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

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

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

**Parameters**

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

**Returns** integral value

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

Fourier analys

**Parameters**

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

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

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

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

## 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  $VALUE \leq 0$

**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  $VALUE \leq 0$

**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*, *numberOfObjectives*, *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*, *theOptimizationVariables*, *theOptimizationAlgorithm*=default['Algo'], *theOptimizationCharacterization*=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 devices

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

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

Evaluates the performances of device associated to entrance vector *x*. Outputs the objective function and the constraints, and other data used in *optiHistoric*.

This function is NOT process safe: “self.” is 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  $VALUE \leq 0$

**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 devices

    set_max_opti_time (self, max_time_sec)

    evaluateObjectiveAndConstraints (self, x)
        Evaluates the performances of device associated to entrance vector x. Outputs the objective function and
        the constraints, and other data used in optiHistoric.

        This function is NOT process safe: “self.” is 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\_measure**

## Module Contents

**class LineItem** (*point1*, *point2*)

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

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

**boundingRect** (*self*)

**class on\_click\_measure**

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

On Click: Measure distance. Click on two points to perform that action

```

graph_clicked (self, the_graph_visual, index_graph, index_trace, indices_points)
reset_distance (self)
display_distance (self)
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)
        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 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 on_click_measure
    Bases: optimeed.visualize.gui.widgets.widget_graphs_visual.on_graph_click_interface
    On Click: Measure distance. Click on two points to perform that action
    graph_clicked (self, the_graph_visual, index_graph, index_trace, indices_points)
    reset_distance (self)
    display_distance (self)
    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)

log_mode (self, x=False, y=False)

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 OnClicSelector:

```

def __init__(self): self.p_list = []

def add_point(self, newp): self.p_list.append(newp)

def draw(self, painter):
    if len(self.p_list) > 2: pen = fn.mkPen(1) pen.setWidthF(2) painter.setPen(pen)
    painter.drawPolyline(QtGui.QPolygonF(self.p_list))

def reset(self): self.p_list = []

def getPath(self): return path.Path([(p.x(), p.y()) for p in self.p_list] + [(self.p_list[-1].x(), self.p_list[-1].y())])

def mouseDragEvent(self, ev):
    if ev.modifiers() and QtCore.Qt.ControlModifier: ev.accept()
    self.clicSelector.add_point(ev.pos()) if ev.isFinish():

    path = self.clicSelector.getPath() points = self.points() contains_points =
    path.contains_points([(p.pos().x(), p.pos().y()) for p in points]) indices = [i for i,
    cond in enumerate(contains_points) if cond] points_clicked = [points[i] for i in
    indices] self.ptsClicked = points_clicked self.sigClicked.emit(self, self.ptsClicked)
    self.clicSelector.reset()

    self.update()

else: ev.ignore()

```

```
class myGraphicsLayoutWidget (parent=None, **_kwargs)
    Bases:      optimeed.visualize.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*)  
Overwriten 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*)  
Overwriten to add font-family to options

**class myAxis** (*orientation*)

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

**get\_label\_pos** (*self*)  
Overwriten to place label closer to the axis

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

**get\_point** (*self*, *indexPoint*)

Return object pyqtgraph.SpotItem

## 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 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**

**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_click_measure
    Bases:                                     optimeed.visualize.gui.widgets.widget_graphs_visual.
                                                on_graph_click_interface
    On Click: Measure distance. Click on two points to perform that action

```

```
graph_clicked (self, the_graph_visual, index_graph, index_trace, indices_points)
reset_distance (self)
display_distance (self)
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: Ac-
tion_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_click_measure
    Bases: optimeed.visualize.gui.widgets.widget_graphs_visual.on_graph_click_interface
    On Click: Measure distance. Click on two points to perform that action
    graph_clicked (self, the_graph_visual, index_graph, index_trace, indices_points)
    reset_distance (self)
    display_distance (self)
    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

```
check_if_must_plot (elem)

class OptimizationDisplayer (thePipeOpti, listOfObjectives, theOptimizer, additionalWid-
                             gets=None)
    Bases: optimeed.core.Option_class

    Class used to display optimization process in real time

    signal_optimization_over
```

```

SHOW_CONSTRAINTS = 0

set_actionsOnClick (self, theList)
    Set actions to perform on click, list of on_graph_click_interface

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

__change_appearance_violate_constraints (self)

__refresh (self)

start_autorefresh (self, timer_autosave)

stop_autorefresh (self)

__set_graphs_disposition (self)
    Set nicely the graphs disposition

launch_optimization (self, refresh_time=0.1)
    Perform the optimization and spawn the convergence graphs afterwards.

__callback_optimization (self)

close_windows (self)

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

```

**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_click_measure
    Bases: optimeed.visualize.gui.widgets.widget_graphs_visual.on_graph_click_interface
    On Click: Measure distance. Click on two points to perform that action
    graph_clicked (self, the_graph_visual, index_graph, index_trace, indices_points)
    reset_distance (self)
    display_distance (self)
    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,  
                      check_if_plot_elem=None)
```

```
    exclude_col (self, id_col)  
        Add id_col to exclude from the graph
```

```
    __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)
```

```
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 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 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

check_if_must_plot (elem)

class OptimizationDisplayer (thePipeOpti, listOfObjectives, theOptimizer, additionalWid-
                             gets=None)
    Bases: optimeed.core.Option_class
    Class used to display optimization process in real time

    signal_optimization_over

    SHOW_CONSTRAINTS = 0

    set_actionsOnClick (self, theList)
        Set actions to perform on click, list of on_graph_click_interface

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

    __change_appearance_violate_constraints (self)

    __refresh (self)

    start_autorefresh (self, timer_autosave)

    stop_autorefresh (self)

    __set_graphs_disposition (self)
        Set nicely the graphs disposition

    launch_optimization (self, refresh_time=0.1)
        Perform the optimization and spawn the convergence graphs afterwards.

    __callback_optimization (self)

    close_windows (self)

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

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 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 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\_symbolsizes** (*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** `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 = 1.1.2
```

### 7.1 Developer 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::*twine upload dist/\**





### O

optimeed, [19](#)  
optimeed consolidate, [19](#)  
optimeed consolidate parametric\_analysis, [19](#)  
optimeed core, [21](#)  
optimeed core additional\_tools, [24](#)  
optimeed core ansi2html, [21](#)  
optimeed core ansi2html converter, [21](#)  
optimeed core ansi2html style, [23](#)  
optimeed core ansi2html util, [23](#)  
optimeed core collection, [25](#)  
optimeed core color\_palette, [27](#)  
optimeed core commonImport, [27](#)  
optimeed core graphs, [27](#)  
optimeed core interfaceDevice, [31](#)  
optimeed core linkDataGraph, [31](#)  
optimeed core myjson, [33](#)  
optimeed core options, [33](#)  
optimeed core tools, [34](#)  
optimeed optimize, [46](#)  
optimeed optimize characterization, [47](#)  
optimeed optimize characterization characterization, [47](#)  
optimeed optimize characterization interfaceCharacterization, [47](#)  
optimeed optimize mathsToPhysics, [47](#)  
optimeed optimize mathsToPhysics interfaceMathsToPhysics, [47](#)  
optimeed optimize mathsToPhysics mathsToPhysics, [48](#)  
optimeed optimize objAndCons, [48](#)  
optimeed optimize objAndCons fastObjCons, [48](#)  
optimeed optimize objAndCons interfaceObjCons, [48](#)  
optimeed optimize optiAlgorithms, [49](#)  
optimeed optimize optiAlgorithms algorithmInterface, [52](#)  
optimeed optimize optiAlgorithms convergence, [49](#)  
optimeed optimize optiAlgorithms convergence evolution, [49](#)  
optimeed optimize optiAlgorithms convergence hyper, [50](#)  
optimeed optimize optiAlgorithms convergence inter, [51](#)  
optimeed optimize optiAlgorithms multiObjective\_GA, [52](#)  
optimeed optimize optiAlgorithms NLOpt\_Algorithm, [52](#)  
optimeed optimize optimizer, [55](#)  
optimeed optimize optiVariable, [54](#)  
optimeed visualize, [60](#)  
optimeed visualize displayOptimization, [96](#)  
optimeed visualize fastPlot, [97](#)  
optimeed visualize gui, [60](#)  
optimeed visualize gui gui\_collection\_exporter, [86](#)  
optimeed visualize gui gui\_data\_animation, [86](#)  
optimeed visualize gui gui\_data\_selector, [88](#)  
optimeed visualize gui gui\_mainWindow, [89](#)  
optimeed visualize gui widgets, [60](#)  
optimeed visualize gui widgets graphsVisualWidget, [60](#)  
optimeed visualize gui widgets graphsVisualWidget, [60](#)  
optimeed visualize gui widgets graphsVisualWidget, [60](#)  
optimeed visualize gui widgets graphsVisualWidget, [61](#)  
optimeed visualize gui widgets graphsVisualWidget, [61](#)  
optimeed visualize gui widgets graphsVisualWidget, [61](#)

```
optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick.on_click_export_co
62
optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick.on_click_extract_p
62
optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick.on_click_measure,
62
optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick.on_click_remove_tr
63
optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick.on_click_showinfo,
63
optimeed.visualize.gui.widgets.graphsVisualWidget.graphVisual,
67
optimeed.visualize.gui.widgets.graphsVisualWidget.pyqtgraphRedefine,
69
optimeed.visualize.gui.widgets.graphsVisualWidget.smallGui,
71
optimeed.visualize.gui.widgets.graphsVisualWidget.traceVisual,
71
optimeed.visualize.gui.widgets.openGLWidget,
73
optimeed.visualize.gui.widgets.openGLWidget.ContextHandler,
73
optimeed.visualize.gui.widgets.openGLWidget.DeviceDrawerInterface,
74
optimeed.visualize.gui.widgets.openGLWidget.Materials_visual,
74
optimeed.visualize.gui.widgets.openGLWidget.OpenGlFunctions_Library,
75
optimeed.visualize.gui.widgets.openGLWidget.quaternions,
76
optimeed.visualize.gui.widgets.openGLWidget.TriangulatePolygon,
76
optimeed.visualize.gui.widgets.widget_graphs_visual,
77
optimeed.visualize.gui.widgets.widget_line_drawer,
78
optimeed.visualize.gui.widgets.widget_menuButton,
79
optimeed.visualize.gui.widgets.widget_openGL,
79
optimeed.visualize.gui.widgets.widget_text,
79
```

## Symbols

- `__CONVERGENCE` (*OptiHistoric* attribute), 56
- `__DATA_STR` (*ListDataStruct* attribute), 26, 37
- `__DEVICE` (*OptiHistoric* attribute), 55
- `__INFO_STR` (*ListDataStruct* attribute), 26, 37
- `__LOGOPTI` (*OptiHistoric* attribute), 55
- `__RESULTS` (*OptiHistoric* attribute), 55
- `__State` (class in *optimeed.core.ansi2html.converter*), 22
- `__amb3__` (*MaterialRenderingProperties* attribute), 75
- `__author__` (in module *optimeed.optimize.optiAlgorithms.convergence.hypervolume*), 50
- `__callback_optimization()` (*OptimizationDisplay* method), 97, 108
- `__change_appearance_violate_constraints()` (*OptimizationDisplay* method), 97, 108
- `__check_graphs()` (*widget\_graphs\_visual* method), 77, 80, 91, 100, 106, 109
- `__create_graph()` (*widget\_graphs\_visual* method), 77, 80, 91, 100, 106, 109
- `__dif3__` (*MaterialRenderingProperties* attribute), 75
- `__draw_axis__()` (*ContextHandler* method), 74
- `__getstate__()` (*AutosaveStruct* method), 26, 37
- `__len__()` (*MultiList* method), 51
- `__lightingInit__()` (*ContextHandler* method), 74
- `__refresh()` (*OptimizationDisplay* method), 97, 108
- `__reset__()` (*ContextHandler* method), 74
- `__set_graphs_disposition()` (*OptimizationDisplay* method), 97, 108
- `__setstate__()` (*AutosaveStruct* method), 26, 37
- `__shin__` (*MaterialRenderingProperties* attribute), 75
- `__spec3__` (*MaterialRenderingProperties* attribute), 75
- `__str__()` (*Attribute\_selector* method), 88
- `__str__()` (*AutosaveStruct* method), 25, 37
- `__str__()` (*Binary\_OptimizationVariable* method), 55, 58
- `__str__()` (*Container\_attribute\_selector* method), 88
- `__str__()` (*Data* method), 29, 41, 112
- `__str__()` (*DataStruct\_Interface* method), 25, 36
- `__str__()` (*HowToPlotGraph* method), 31, 43, 106
- `__str__()` (*Integer\_OptimizationVariable* method), 55, 59
- `__str__()` (*InterfaceCharacterization* method), 47, 57
- `__str__()` (*InterfaceObjCons* method), 49, 58
- `__str__()` (*MathsToPhysics* method), 48, 57
- `__str__()` (*MultiList* method), 50
- `__str__()` (*MultiList.Node* method), 50
- `__str__()` (*MultiObjective\_GA* method), 53, 54, 58
- `__str__()` (*NLOpt\_Algorithm* method), 52
- `__str__()` (*OptimizationVariable* method), 54
- `__str__()` (*Options* method), 34, 45
- `__str__()` (*Real\_OptimizationVariable* method), 54, 58
- `__str__()` (*Rule* method), 23
- `__apply_regex()` (*Ansi2HTMLConverter* method), 23, 24
- `__collapse_cursor()` (*Ansi2HTMLConverter* method), 23, 24
- `__find_class()` (in module *optimeed.core.myjson*), 33
- `__get_annotations()` (in module *optimeed.core.myjson*), 33
- `__get_object_class()` (in module *optimeed.core.myjson*), 33
- `__get_object_module()` (in module *optimeed.core.myjson*), 33
- `__html_template` (in module *optimeed.core.ansi2html.converter*), 22
- `__instantiates_annotated_object()` (in module *optimeed.core.myjson*), 33
- `__latex_template` (in module *optimeed.core.ansi2html.converter*), 22
- `__needs_extra_newline()` (in module *optimeed.core.ansi2html.converter*), 22
- `__object_to_FQCN()` (in module *optimeed.core.myjson*), 33
- `__remove_index_from_show()` (*DataAnimationTrace* method), 87

## A

- Action\_on\_selector\_update (class in *optimeed.visualize.gui.gui\_data\_selector*), 88
- activateMaterialProperties() (*MaterialRenderingProperties* method), 75
- add\_attribute\_selector() (*Container\_attribute\_selector* method), 88
- add\_child() (*Attribute\_selector* method), 88
- add\_child() (*Container\_attribute\_selector* method), 88
- add\_collection() (*LinkDataGraph* method), 32, 44, 105
- add\_data() (*GraphVisual* method), 68
- add\_data() (*ListDataStruct* method), 26, 37
- add\_data\_to\_collection() (*gui\_collection\_exporter* method), 86, 90, 99
- add\_element() (*DataAnimationTrace* method), 87
- add\_elementToTrace() (*DataAnimationVisuals* method), 65, 84, 87, 90, 95, 99
- add\_feature() (*GraphVisual* method), 68
- add\_graph() (*Graphs* method), 30, 43, 113
- add\_graph() (*LinkDataGraph* method), 32, 44, 105
- add\_index\_to\_show() (*DataAnimationTrace* method), 87
- add\_link() (*LinkDataGraph.collection\_linker* method), 32, 44, 105
- add\_modified\_paintElem() (*TraceVisual.ModifiedPaintElem* method), 71
- add\_option() (*Option* class method), 20, 34, 45, 106
- add\_option() (*Options* method), 34, 45
- add\_plot() (*PlotHolders* method), 97, 113
- add\_plot() (*WindowHolders* method), 97, 114
- add\_point() (*ConvergenceManager* method), 52
- add\_point() (*Data* method), 29, 41, 112
- add\_point() (*OptiHistoric* method), 56
- add\_suffix\_to\_path() (in module *optimeed.core*), 39
- add\_suffix\_to\_path() (in module *optimeed.core.tools*), 35
- add\_trace() (*DataAnimationVisuals* method), 65, 84, 87, 90, 95, 99
- add\_trace() (*Graph* method), 29, 42
- add\_trace() (*Graphs* method), 30, 42, 112
- add\_trace() (*GraphVisual* method), 68
- add\_trace\_firstGraph() (*Graphs* method), 30, 42, 112
- add\_update\_method() (*Graphs* method), 31, 43, 113
- add\_visual() (*on\_graph\_click\_showInfo.DataInformationVisuals* method), 63, 64, 83, 94, 103, 108
- addItem() (*myGraphicsLayout* method), 70
- addItem() (*myLegend* method), 70
- adjust() (*\_State* method), 22
- ALGORITHM (*NLOpt\_Algorithm* attribute), 52
- AlgorithmInterface (class in *optimeed.optimize.optiAlgorithms.algorithmInterface*), 52
- Ansi2HTMLConverter (class in *optimeed.core.ansi2html*), 24
- Ansi2HTMLConverter (class in *optimeed.core.ansi2html.converter*), 22
- ANSI\_BACKGROUND\_256 (in module *optimeed.core.ansi2html.converter*), 22
- ANSI\_BACKGROUND\_CUSTOM\_MAX (in module *optimeed.core.ansi2html.converter*), 22
- ANSI\_BACKGROUND\_CUSTOM\_MIN (in module *optimeed.core.ansi2html.converter*), 22
- ANSI\_BACKGROUND\_DEFAULT (in module *optimeed.core.ansi2html.converter*), 22
- ANSI\_BACKGROUND\_HIGH\_INTENSITY\_MAX (in module *optimeed.core.ansi2html.converter*), 22
- ANSI\_BACKGROUND\_HIGH\_INTENSITY\_MIN (in module *optimeed.core.ansi2html.converter*), 22
- ANSI\_BLINK\_FAST (in module *optimeed.core.ansi2html.converter*), 21
- ANSI\_BLINK\_OFF (in module *optimeed.core.ansi2html.converter*), 21
- ANSI\_BLINK\_SLOW (in module *optimeed.core.ansi2html.converter*), 21
- ANSI\_CROSSED\_OUT\_OFF (in module *optimeed.core.ansi2html.converter*), 21
- ANSI\_CROSSED\_OUT\_ON (in module *optimeed.core.ansi2html.converter*), 21
- ANSI\_FOREGROUND\_256 (in module *optimeed.core.ansi2html.converter*), 22
- ANSI\_FOREGROUND\_CUSTOM\_MAX (in module *optimeed.core.ansi2html.converter*), 22
- ANSI\_FOREGROUND\_CUSTOM\_MIN (in module *optimeed.core.ansi2html.converter*), 21
- ANSI\_FOREGROUND\_DEFAULT (in module *optimeed.core.ansi2html.converter*), 22
- ANSI\_FOREGROUND\_HIGH\_INTENSITY\_MAX (in module *optimeed.core.ansi2html.converter*), 22
- ANSI\_FOREGROUND\_HIGH\_INTENSITY\_MIN (in module *optimeed.core.ansi2html.converter*), 22
- ANSI\_FULL\_RESET (in module *optimeed.core.ansi2html.converter*), 21
- ANSI\_INTENSITY\_INCREASED (in module *optimeed.core.ansi2html.converter*), 21
- ANSI\_INTENSITY\_NORMAL (in module *optimeed.core.ansi2html.converter*), 21
- ANSI\_INTENSITY\_REDUCED (in module *optimeed.core.ansi2html.converter*), 21
- ANSI\_NEGATIVE\_OFF (in module *optimeed.core.ansi2html.converter*), 22
- ANSI\_NEGATIVE\_ON (in module *optimeed.core.ansi2html.converter*), 22

- ANSI\_STYLE\_ITALIC (in module *optimeed.core.ansi2html.converter*), 21
- ANSI\_STYLE\_NORMAL (in module *optimeed.core.ansi2html.converter*), 21
- ANSI\_UNDERLINE\_OFF (in module *optimeed.core.ansi2html.converter*), 21
- ANSI\_UNDERLINE\_ON (in module *optimeed.core.ansi2html.converter*), 21
- ANSI\_VISIBILITY\_OFF (in module *optimeed.core.ansi2html.converter*), 21
- ANSI\_VISIBILITY\_ON (in module *optimeed.core.ansi2html.converter*), 21
- app (in module *optimeed.visualize*), 98
- app (in module *optimeed.visualize.gui*), 89
- app (in module *optimeed.visualize.gui.gui\_data\_selector*), 88
- app (in module *optimeed.visualize.gui.gui\_mainWindow*), 89
- append() (*MultiList* method), 51
- apply() (*on\_graph\_click\_delete* method), 62, 64, 82, 93, 102
- apply\_filters() (*GuiDataSelector* method), 88
- apply\_palette() (*GraphVisual* method), 68
- apply\_regex() (*Ansi2HTMLConverter* method), 23, 24
- apply\_width\_sample() (*myLegend* method), 70
- applyEquation() (in module *optimeed.core*), 36, 38
- applyEquation() (in module *optimeed.core.tools*), 34
- arithmeticEval() (in module *optimeed.core*), 39
- arithmeticEval() (in module *optimeed.core.tools*), 35
- assign() (*InterfaceDevice* method), 31, 43
- Attribute\_selector (class in *optimeed.visualize.gui.gui\_data\_selector*), 88
- attrs() (*Ansi2HTMLConverter* method), 23, 24
- AutosaveStruct (class in *optimeed.core*), 36
- AutosaveStruct (class in *optimeed.core.collection*), 25
- axis\_equal() (*GraphVisual* method), 69
- axis\_equal() (in module *optimeed.visualize*), 114
- axis\_equal() (in module *optimeed.visualize.fastPlot*), 98
- axis\_equal() (*PlotHolders* method), 97, 113
- axis\_equal() (*WindowHolders* method), 98, 114
- axisangle\_to\_q() (in module *optimeed.visualize.gui.widgets.openGLWidget.quaternions*), 76
- B**
- Binary\_OptimizationVariable (class in *optimeed.optimize*), 58
- Binary\_OptimizationVariable (class in *optimeed.optimize.optiVariable*), 55
- blackOnly() (in module *optimeed.core*), 38
- blackOnly() (in module *optimeed.core.color\_palette*), 27
- BLUE (*text\_format* attribute), 20, 34, 38, 45
- Blue\_material (in module *optimeed.visualize.gui.widgets.openGLWidget.Materials\_visual*), 75
- BOLD (*text\_format* attribute), 20, 34, 38, 45
- boundingRect() (*LineItem* method), 62
- Brass\_material (in module *optimeed.visualize.gui.widgets.openGLWidget.Materials\_visual*), 75
- Bronze\_material (in module *optimeed.visualize.gui.widgets.openGLWidget.Materials\_visual*), 75
- C**
- callback\_on\_evaluation() (*Optimizer* method), 57, 59
- cart2pol() (in module *optimeed.core*), 46
- cart2pol() (in module *optimeed.core.additional\_tools*), 24
- Characterization (class in *optimeed.optimize*), 57
- Characterization (class in *optimeed.optimize.characterization*), 47
- Characterization (class in *optimeed.optimize.characterization.characterization*), 47
- check\_and\_add\_if\_float() (in module *optimeed.visualize.gui.gui\_data\_selector*), 88
- check\_if\_must\_plot() (in module *optimeed.visualize*), 108
- check\_if\_must\_plot() (in module *optimeed.visualize.displayOptimization*), 96
- Chrome\_material (in module *optimeed.visualize.gui.widgets.openGLWidget.Materials\_visual*), 75
- CLASS\_TAG (in module *optimeed.core.myjson*), 33
- CLIC\_LEFT (in module *optimeed.visualize.gui.widgets.openGLWidget.ContextHandler*), 74
- CLIC\_RIGHT (in module *optimeed.visualize.gui.widgets.openGLWidget.ContextHandler*), 74
- close() (*MyMultiprocessEvaluator* method), 53
- close\_windows() (*OptimizationDisplayer* method), 97, 108
- closeEvent() (*DataAnimationVisuals* method), 66, 84, 88, 90, 95, 99
- closeEvent() (*gui\_mainWindow* method), 89, 98, 106, 110
- CollectionInfo (class in *optimeed.core*), 43
- CollectionInfo (class in *optimeed.core.linkDataGraph*), 31



`color()` (in module *optimeed.core.ansi2html.style*), 23  
`color_component()` (in module *optimeed.core.ansi2html.style*), 23  
`compute()` (*Characterization method*), 47, 57  
`compute()` (*FastObjCons method*), 48, 49, 57  
`compute()` (*HyperVolume method*), 50  
`compute()` (*MultiObjective\_GA method*), 53, 54, 58  
`compute()` (*NLOpt\_Algorithm method*), 52  
`constraints` (*OptiHistoric.\_pointData attribute*), 55  
`constraints_per_step` (*EvolutionaryConvergence attribute*), 49, 51  
`Container_attribute_selector` (class in *optimeed.visualize.gui.gui\_data\_selector*), 88  
`contains_trace()` (*DataAnimationVisuals method*), 66, 84, 88, 90, 95, 99  
`ContextHandler` (class in *optimeed.visualize.gui.widgets.openGLWidget.ContextHandler*), 66, 74  
`conv` (*MyConvergence attribute*), 52  
`ConvergenceManager` (class in *optimeed.optimize.optiAlgorithms.NLOpt\_Algorithm*), 52  
`convert()` (*Ansi2HTMLConverter method*), 23, 24  
`Copper_material` (in module *optimeed.visualize.gui.widgets.openGLWidget.MaterialsVisual*), 75  
`copy()` (*Options method*), 34, 45  
`create_main_window()` (*OptimizationDisplayer method*), 97, 108  
`create_trace()` (*LinkDataGraph method*), 32, 44, 105  
`create_unique_dirname()` (in module *optimeed.core*), 38  
`create_unique_dirname()` (in module *optimeed.core.tools*), 34  
`createGraphs()` (*LinkDataGraph method*), 32, 44, 105  
`curr_index()` (*on\_graph\_click\_showInfo.DataInformationVisuals method*), 63, 64, 83, 94, 103, 108  
`CursorMoveUp` (class in *optimeed.core.ansi2html.converter*), 22  
`CYAN` (*text\_format attribute*), 20, 34, 38, 45

## D

`dark2()` (in module *optimeed.core*), 38  
`dark2()` (in module *optimeed.core.color\_palette*), 27  
`DARKCYAN` (*text\_format attribute*), 20, 34, 38, 45  
`Data` (class in *optimeed.core*), 40  
`Data` (class in *optimeed.core.graphs*), 27  
`Data` (class in *optimeed.visualize*), 110  
`DataAnimationLines` (class in *optimeed.visualize*), 104  
`DataAnimationLines` (class in *optimeed.visualize.gui*), 96  
`DataAnimationLines` (class in *optimeed.visualize.gui.widgets*), 85  
`DataAnimationLines` (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionO*), 66  
`DataAnimationLines` (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionO*), 60  
`DataAnimationOpenGL` (class in *optimeed.visualize*), 104  
`DataAnimationOpenGL` (class in *optimeed.visualize.gui*), 95  
`DataAnimationOpenGL` (class in *optimeed.visualize.gui.widgets*), 85  
`DataAnimationOpenGL` (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionO*), 66  
`DataAnimationOpenGL` (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionO*), 60  
`DataAnimationOpenGLwText` (class in *optimeed.visualize*), 104  
`DataAnimationOpenGLwText` (class in *optimeed.visualize.gui*), 96  
`DataAnimationOpenGLwText` (class in *optimeed.visualize.gui.widgets*), 85  
`DataAnimationOpenGLwText` (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionO*), 66  
`DataAnimationOpenGLwText` (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionO*), 60  
`DataAnimationTrace` (class in *optimeed.visualize.gui.gui\_data\_animation*), 86  
`DataAnimationTrace.element_animation` (class in *optimeed.visualize.gui.gui\_data\_animation*), 86  
`DataAnimationVisuals` (class in *optimeed.visualize*), 99  
`DataAnimationVisuals` (class in *optimeed.visualize.gui*), 90, 95  
`DataAnimationVisuals` (class in *optimeed.visualize.gui.gui\_data\_animation*), 87  
`DataAnimationVisuals` (class in *optimeed.visualize.gui.widgets*), 84  
`DataAnimationVisuals` (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionO*), 65  
`DataAnimationVisualswText` (class in *optimeed.visualize*), 104  
`DataAnimationVisualswText` (class in *optimeed.visualize.gui*), 96

- meed.visualize.gui*), 96
- DataAnimationVisualswText (class in *optimeed.visualize.gui.widgets*), 85
- DataAnimationVisualswText (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick*), 67
- DataAnimationVisualswText (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick*), 60
- DataStruct\_Interface (class in *optimeed.core*), 36
- DataStruct\_Interface (class in *optimeed.core.collection*), 25
- decode\_str\_json() (in module *optimeed.core*), 36
- decode\_str\_json() (in module *optimeed.core.myjson*), 33
- default (in module *optimeed.optimize.optimizer*), 55
- default\_palette() (in module *optimeed.core*), 38
- default\_palette() (in module *optimeed.core.color\_palette*), 27
- delete() (*GraphVisual* method), 69
- delete() (*widget\_graphs\_visual* method), 77, 80, 91, 100, 107, 109
- delete\_all() (*DataAnimationTrace* method), 87
- delete\_all() (*DataAnimationVisuals* method), 65, 84, 87, 90, 95, 99
- delete\_graph() (*widget\_graphs\_visual* method), 77, 80, 91, 100, 107, 109
- delete\_gui (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick*), 61
- delete\_indices\_from\_list() (in module *optimeed.core*), 39
- delete\_indices\_from\_list() (in module *optimeed.core.tools*), 35
- delete\_key\_widgets() (*DataAnimationLines* method), 60, 67, 85, 96, 104
- delete\_key\_widgets() (*DataAnimationOpenGL* method), 60, 66, 85, 96, 104
- delete\_lines() (*widget\_line\_drawer* method), 66, 78, 81, 85, 92, 101
- delete\_point() (*Data* method), 29, 41, 112
- delete\_point() (*DataAnimationVisuals* method), 65, 84, 87, 90, 95, 99
- delete\_points\_at\_indices() (*ListDataStruct* method), 26, 38
- delete\_trace() (*GraphVisual* method), 68
- delete\_visual() (*on\_graph\_click\_showInfo.DataInformationVisual* method), 63, 64, 83, 94, 103, 108
- derivate() (in module *optimeed.core*), 46
- derivate() (in module *optimeed.core.additional\_tools*), 24
- DeviceDrawerInterface (class in *optimeed.visualize*), 102
- DeviceDrawerInterface (class in *optimeed.visualize.gui*), 93
- DeviceDrawerInterface (class in *optimeed.visualize.gui.widgets*), 86
- DeviceDrawerInterface (class in *optimeed.visualize.gui.widgets.openGLWidget.DeviceDrawerInterface*), 74
- DeviceDrawerInterface (class in *optimeed.visualize.gui.widgets.openGLWidget.DeviceDrawerInterface*), 63, 67, 84, 95, 104
- DISPLAY\_INFO (*Optimizer* attribute), 56, 59
- dist() (in module *optimeed.core*), 46
- dist() (in module *optimeed.core.additional\_tools*), 24
- DIVISION\_OUTER (*MultiObjective\_GA* attribute), 53, 54, 58
- do\_MathsToPhys() (*Binary\_OptimizationVariable* method), 55, 58
- do\_MathsToPhys() (*Integer\_OptimizationVariable* method), 55, 59
- do\_MathsToPhys() (*OptimizationVariable* method), 54
- do\_MathsToPhys() (*Real\_OptimizationVariable* method), 54, 58
- draw\_2Dring() (in module *optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctions\_Lib*), 76
- draw\_2Dring\_diff\_angle() (in module *optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctions\_Lib*), 76
- draw\_carved\_disk() (in module *optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctions\_Lib*), 76
- draw\_closedPolygon() (in module *optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctions\_Lib*), 75
- draw\_cylinder() (in module *optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctions\_Lib*), 76
- draw\_disk() (in module *optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctions\_Lib*), 76
- draw\_extrudeZ() (in module *optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctions\_Lib*), 75
- draw\_lines() (in module *optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctions\_Lib*), 75
- draw\_part\_cylinder() (in module *optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctions\_Lib*), 76
- draw\_part\_cylinder\_throat() (in module *optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctions\_Lib*), 76
- draw\_part\_disk() (in module *optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctions\_Lib*), 76

76  
draw\_part\_disk\_diff\_angles() (in module optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctionsLibrary), 76  
draw\_rectangle() (in module optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctionsLibrary), 75  
draw\_simple\_rectangle() (in module optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctionsLibrary), 75  
draw\_spiral() (in module optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctionsLibrary), 75  
draw\_spiralFront() (in module optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctionsLibrary), 75  
draw\_spiralFull() (in module optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctionsLibrary), 75  
draw\_spiralSheet() (in module optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctionsLibrary), 75  
draw\_triList() (in module optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctionsLibrary), 75  
draw\_tubeSheet() (in module optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctionsLibrary), 76  
drawWireTube() (in module optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctionsLibrary), 76  
**E**  
Emerald\_material (in module optimeed.visualize.gui.widgets.openGLWidget.MaterialsVisual), 75  
encode\_str\_json() (in module optimeed.core), 36  
encode\_str\_json() (in module optimeed.core.myjson), 33  
END (text\_format attribute), 20, 34, 38, 45  
evaluate() (MyProblem method), 53  
evaluate() (Parametric\_analysis method), 20, 21  
evaluate\_all() (MyMapEvaluator method), 53  
evaluate\_all() (MyMultiprocessEvaluator method), 53  
evaluateObjectiveAndConstraints() (Optimizer method), 56, 59  
EvolutionaryConvergence (class in optimeed.optimize.optiAlgorithms.convergence), 51  
EvolutionaryConvergence (class in optimeed.optimize.optiAlgorithms.convergence.evolutionaryConvergence), 49  
exclude\_col() (HowToPlotGraph method), 31, 43, 106  
EXCLUDED\_TAGS (in module optimeed.core.myjson), 33  
ExportData() (DataAnimationVisuals method), 66, 84, 88, 90, 95, 99  
export\_str() (Data method), 29, 41, 112  
export\_str() (Graph method), 30, 42  
export\_str() (Graphs method), 31, 43, 113  
export\_widget() (DataAnimationLines method), 60, 67, 85, 96, 104  
export\_widget() (DataAnimationOpenGL method), 60, 66, 85, 96, 104  
ExportXListDataStruct (in module optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctionsLibrary), 27, 38  
exportCollection() (gui\_collection\_exporter method), 86, 90, 98  
ExportGraphs (in module optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctionsLibrary), (widget\_graphs\_visual method), 77, 80, 91, 100, 107, 109  
extend() (MultiList method), 51  
ExtendListDataStruct (in module optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctionsLibrary), (MyConvergence method), 53  
**F**  
FastLUTInterpolation (class in optimeed.core), 45  
fast\_LUT\_interpolation (class in optimeed.core.additional\_tools), 24  
fast\_update() (GraphVisual method), 69  
fast\_update() (widget\_graphs\_visual method), 77, 80, 91, 100, 107, 109  
FastObjCons (class in optimeed.optimize), 57  
FastObjCons (class in optimeed.optimize.objAndCons), 49  
FastObjCons (class in optimeed.optimize.objAndCons.fastObjCons), 48  
figure() (in module optimeed.visualize), 114  
figure\_visual() (in module optimeed.visualize.fastPlot), 98  
find\_and\_replace() (in module optimeed.core), 38  
find\_and\_replace() (in module optimeed.core.tools), 34  
formatInfo() (Optimizer method), 57, 59  
frame\_selector() (DataAnimationVisuals method), 66, 84, 87, 90, 95, 99  
fromMathsToPhys() (MathsToPhysics method), 48, 57  
fromPhysToMaths() (MathsToPhysics method), 48, 57  
**G**  
generate() (MyGenerator method), 53  
generate\_optimizationGraphs() (OptimizationDisplayer method), 97, 108  
get() (DataAnimationTrace.element\_animation method), 87  
get\_2D\_pareto() (in module optimeed.core), 39



`get_2D_pareto()` (in module *optimeed.core.tools*), 35  
`get_all_figures()` (in module *optimeed.visualize*), 114  
`get_all_figures()` (in module *optimeed.visualize.fastPlot*), 98  
`get_all_figures()` (*WindowHolders* method), 98, 114  
`get_all_graphs()` (*Graphs* method), 30, 43, 113  
`get_all_graphs_ids()` (*Graphs* method), 30, 43, 113  
`get_all_graphsVisual()` (*widget\_graphs\_visual* method), 77, 81, 91, 100, 107, 109  
`get_all_id_graphs()` (*LinkDataGraph* method), 32, 44, 105  
`get_all_options()` (*Option\_class* method), 20, 34, 45, 106  
`get_all_traces()` (*Graph* method), 30, 42  
`get_all_traces()` (*GraphVisual* method), 68  
`get_all_traces_id_graph()` (*LinkDataGraph* method), 32, 44, 105  
`get_analyzed_attribute()` (*Parameteric\_parameter* method), 19, 21  
`get_attr_object()` (in module *optimeed.visualize.gui.gui\_data\_selector*), 89  
`get_attribute_name()` (*OptimizationVariable* method), 54  
`get_attribute_selectors()` (*Container\_attribute\_selector* method), 88  
`get_axis()` (*GraphVisual* method), 68  
`get_base_pen()` (*DataAnimationTrace* method), 87  
`get_base_pen()` (*TraceVisual* method), 72  
`get_base_symbol()` (*TraceVisual* method), 72  
`get_base_symbol_brush()` (*TraceVisual* method), 72  
`get_base_symbol_pen()` (*TraceVisual* method), 72  
`get_brushes()` (*TraceVisual* method), 72  
`get_children()` (*Attribute\_selector* method), 88  
`get_children()` (*Container\_attribute\_selector* method), 88  
`get_collection()` (*CollectionInfo* method), 31, 43  
`get_collection_from_graph()` (*LinkData-Graph* method), 32, 44, 105  
`get_collection_master()` (*LinkData-Graph\_collection\_linker* method), 32, 44, 105  
`get_collectionInfo()` (*LinkDataGraph* method), 32, 44, 105  
`get_color()` (*Data* method), 28, 40, 111  
`get_color()` (*TraceVisual* method), 72  
`get_colour_background()` (*DeviceDrawerInterface* method), 74, 86, 93, 102  
`get_colour_scalebar()` (*DeviceDrawerInterface* method), 74, 86, 93, 102  
`get_convergence()` (*MultiObjective\_GA* method), 53, 54, 58  
`get_convergence()` (*NLOpt\_Algorithm* method), 52  
`get_convergence()` (*OptiHistoric* method), 56  
`get_curr_plotHolder()` (*WindowHolders* method), 97, 114  
`get_data()` (*ListDataStruct* method), 26, 37  
`get_data()` (*TraceVisual* method), 72  
`get_dataIndex_from_graphIndex()` (*Data* method), 28, 40, 111  
`get_dataIndices_from_graphIndices()` (*Data* method), 28, 40, 111  
`getDataObject_from_graph()` (*LinkData-Graph* method), 32, 44, 105  
`getDataObjects_from_graph()` (*LinkData-Graph* method), 32, 44, 105  
`get_datastruct()` (*AutosaveStruct* method), 26, 37  
`get_device()` (*PipeOptimization* method), 55  
`get_devices()` (*OptiHistoric* method), 56  
`get_element_animations()` (*DataAnimation-Trace* method), 87  
`get_ellipse_axes()` (in module *optimeed.core*), 46  
`get_ellipse_axes()` (in module *optimeed.core.additional\_tools*), 25  
`get_extrema_lines()` (*widget\_line\_drawer* method), 66, 78, 81, 85, 92, 101  
`get_filename()` (*AutosaveStruct* method), 25, 37  
`get_first_graph()` (*Graphs* method), 30, 42, 113  
`get_graph()` (*Graphs* method), 30, 42, 113  
`get_graph()` (*widget\_graphs\_visual* method), 77, 80, 91, 100, 107, 109  
`get_graph_and_trace_from_collection()` (*LinkDataGraph* method), 32, 44, 105  
`get_graphIndex_from_dataIndex()` (*Data* method), 28, 41, 111  
`get_graphIndices_from_dataIndices()` (*Data* method), 28, 41, 111  
`get_graphs()` (*EvolutionaryConvergence* method), 50, 52  
`get_graphs()` (*MyConvergence* method), 53  
`get_historic()` (*PipeOptimization* method), 55  
`get_howToPlotGraph()` (*LinkDataGraph* method), 32, 44, 105  
`get_hypervolume_convergence()` (*EvolutionaryConvergence* method), 50, 51  
`get_id()` (*CollectionInfo* method), 31, 44  
`get_idCollection_from_graph()` (*LinkData-Graph* method), 32, 44, 105  
`get_indices_points_to_plot()` (*Data* method), 29, 41, 112  
`get_indices_to_show()` (*DataAnimationTrace* method), 87  
`get_info()` (*DataStruct\_Interface* method), 25, 36  
`get_interesting_elements()` (*DataAnimation-*

*Lines method*), 60, 67, 85, 96, 104  
get\_interesting\_elements() (*DataAnimationOpenGLwText method*), 60, 66, 85, 96, 104  
get\_invert\_permutations() (*Data method*), 28, 40, 111  
get\_kwargs() (*CollectionInfo method*), 31, 44  
get\_label\_pos() (*myAxis method*), 71  
get\_last\_pareto() (*EvolutionaryConvergence method*), 49, 51  
get\_layout\_buttons() (*widget\_graphs\_visual method*), 78, 81, 91, 100, 107, 109  
get\_legend() (*Data method*), 29, 41, 112  
get\_legend() (*GraphVisual method*), 68  
get\_length() (*TraceVisual method*), 72  
get\_length\_data() (*Data method*), 28, 40, 110  
get\_linestyle() (*Data method*), 29, 41, 112  
get\_list\_attributes() (*ListDataStruct method*), 26, 37  
get\_logopti() (*OptiHistoric method*), 56  
get\_mappingData\_graph() (*LinkDataGraph method*), 32, 45, 105  
get\_mappingData\_trace() (*LinkDataGraph method*), 32, 45, 105  
get\_max\_value() (*Integer\_OptimizationVariable method*), 55, 58  
get\_max\_value() (*Real\_OptimizationVariable method*), 54, 58  
get\_min\_max\_attributes() (*Attribute\_selector method*), 88  
get\_min\_value() (*Integer\_OptimizationVariable method*), 55, 58  
get\_min\_value() (*Real\_OptimizationVariable method*), 54, 58  
get\_nadir\_point() (*EvolutionaryConvergence method*), 50, 51  
get\_nadir\_point\_all\_steps() (*EvolutionaryConvergence method*), 50, 51  
get\_name() (*Attribute\_selector method*), 88  
get\_name() (*Container\_attribute\_selector method*), 88  
get\_name() (*FastObjCons method*), 48, 49, 57  
get\_name() (*InterfaceObjCons method*), 49, 58  
get\_name() (*on\_click\_change\_symbol method*), 61, 65, 83, 94, 103  
get\_name() (*on\_click\_copy\_something method*), 61, 65, 83, 94, 103  
get\_name() (*on\_click\_extract\_pareto method*), 62, 64, 82, 94, 103  
get\_name() (*on\_click\_measure method*), 63, 67, 84, 95, 104  
get\_name() (*on\_graph\_click\_delete method*), 62, 64, 82, 93, 102  
get\_name() (*on\_graph\_click\_export method*), 62, 64, 82, 93, 102  
get\_name() (*on\_graph\_click\_remove\_trace method*), 63, 65, 83, 94, 103  
get\_name() (*on\_graph\_click\_showAnim method*), 61, 67, 86, 96, 104  
get\_name() (*on\_graph\_click\_showInfo method*), 63, 64, 83, 94, 103, 108  
get\_name() (*Options method*), 33, 45  
get\_nb\_objectives() (*EvolutionaryConvergence method*), 50, 51  
get\_nb\_steps() (*EvolutionaryConvergence method*), 50, 51  
get\_ND\_pareto() (*in module optimeed.core*), 39  
get\_ND\_pareto() (*in module optimeed.core.tools*), 35  
get\_new\_index() (*on\_graph\_click\_showInfo.DataInformationVisuals method*), 63, 64, 83, 94, 103, 108  
get\_number\_of\_elements() (*DataAnimationTrace method*), 87  
get\_number\_of\_points() (*Data method*), 28, 40, 111  
get\_object\_attrs() (*in module optimeed.core*), 39  
get\_object\_attrs() (*in module optimeed.core.tools*), 35  
get\_opengl\_options() (*DeviceDrawerInterface method*), 74, 86, 93, 102  
get\_optionValue() (*Option\_class method*), 20, 34, 45, 106  
get\_pareto\_convergence() (*EvolutionaryConvergence method*), 49, 51  
get\_permutations() (*Data method*), 28, 40, 111  
get\_PhysToMaths() (*Binary\_OptimizationVariable method*), 55, 58  
get\_PhysToMaths() (*Integer\_OptimizationVariable method*), 55, 59  
get\_PhysToMaths() (*OptimizationVariable method*), 54  
get\_PhysToMaths() (*Real\_OptimizationVariable method*), 54, 58  
get\_plot\_data() (*Data method*), 28, 40, 111  
get\_point() (*TraceVisual method*), 73  
get\_population\_size() (*EvolutionaryConvergence method*), 50, 52  
get\_reference\_device() (*Parametric\_parameter method*), 19, 20  
get\_results() (*OptiHistoric method*), 56  
get\_styles() (*in module optimeed.core.ansi2html.style*), 23  
get\_symbol() (*Data method*), 29, 41, 112  
get\_symbol() (*TraceVisual method*), 72  
get\_symbolOutline() (*Data method*), 28, 40, 110  
get\_symbolPens() (*TraceVisual method*), 73  
get\_symbolsSize() (*Data method*), 28, 40, 110  
get\_text\_to\_write() (*ContextHandler method*),

- 74
- `get_trace()` (*Graph method*), 29, 42
- `get_trace()` (*GraphVisual method*), 68
- `get_value()` (*Options method*), 33, 45
- `get_values()` (*Parametric\_minmax method*), 19, 21
- `get_wgGraphs()` (*in module optimeed.visualize*), 114
- `get_wgGraphs()` (*in module optimeed.visualize.fastPlot*), 98
- `get_wgGraphs()` (*PlotHolders method*), 97, 113
- `get_wgGraphs()` (*WindowHolders method*), 97, 114
- `get_widget()` (*Repr\_lines method*), 63, 65, 83, 94, 103
- `get_widget()` (*Repr\_opengl method*), 63, 64, 83, 94, 103
- `get_width()` (*Data method*), 28, 40, 111
- `get_x()` (*Data method*), 27, 40, 110
- `get_x_label()` (*Data method*), 29, 41, 112
- `get_xlim()` (*Data method*), 28, 40, 111
- `get_y()` (*Data method*), 28, 40, 111
- `get_y_label()` (*Data method*), 29, 41, 112
- `get_ylim()` (*Data method*), 28, 40, 111
- `getAmb3()` (*MaterialRenderingProperties method*), 75
- `getDif3()` (*MaterialRenderingProperties method*), 75
- `GetEar()` (*in module optimeed.visualize.gui.widgets.openGLWidget.TriangularPolygon*), 76
- `getLength()` (*MultiList method*), 51
- `getLineInfo()` (*in module optimeed.core*), 39
- `getLineInfo()` (*in module optimeed.core.tools*), 35
- `getPath_workspace()` (*in module optimeed consolidate*), 20
- `getPath_workspace()` (*in module optimeed.core*), 35, 39
- `getPath_workspace()` (*in module optimeed.core.tools*), 35
- `getShin()` (*MaterialRenderingProperties method*), 75
- `getSpec3()` (*MaterialRenderingProperties method*), 75
- `Graph` (*class in optimeed.core*), 41
- `Graph` (*class in optimeed.core.graphs*), 29
- `graph_clicked()` (*on\_click\_change\_symbol method*), 61, 65, 83, 94, 103
- `graph_clicked()` (*on\_click\_copy\_something method*), 61, 65, 83, 94, 103
- `graph_clicked()` (*on\_click\_extract\_pareto method*), 62, 64, 82, 94, 103
- `graph_clicked()` (*on\_click\_measure method*), 62, 67, 83, 95, 104
- `graph_clicked()` (*on\_graph\_click\_delete method*), 62, 64, 82, 93, 102
- `graph_clicked()` (*on\_graph\_click\_export method*), 62, 64, 82, 93, 102
- `graph_clicked()` (*on\_graph\_click\_remove\_trace method*), 63, 65, 83, 94, 103
- `graph_clicked()` (*on\_graph\_click\_showAnim method*), 61, 67, 86, 96, 104
- `graph_clicked()` (*on\_graph\_click\_showInfo method*), 63, 64, 83, 94, 103, 108
- `Graphs` (*class in optimeed.core*), 42
- `Graphs` (*class in optimeed.core.graphs*), 30
- `Graphs` (*class in optimeed.visualize*), 112
- `GraphVisual` (*class in optimeed.visualize.gui.widgets.graphsVisualWidget.graphVisual*), 67
- `GREEN` (*text\_format attribute*), 20, 34, 38, 45
- `grid_off()` (*GraphVisual method*), 69
- `gui_collection_exporter` (*class in optimeed.visualize*), 98
- `gui_collection_exporter` (*class in optimeed.visualize.gui*), 89
- `gui_collection_exporter` (*class in optimeed.visualize.gui.gui\_collection\_exporter*), 86
- `gui_mainWindow` (*class in optimeed.visualize*), 98, 106, 110
- `gui_mainWindow` (*class in optimeed.visualize.gui*), 89
- `gui_mainWindow` (*class in optimeed.visualize.gui.gui\_mainWindow*), 89
- `gui_data_selector` (*class in optimeed.visualize.gui.gui\_data\_selector*), 88
- `guiPyqtgraph` (*class in optimeed.visualize*), 102, 108, 113
- `guiPyqtgraph` (*class in optimeed.visualize.gui*), 93
- `guiPyqtgraph` (*class in optimeed.visualize.gui.widgets*), 86
- `guiPyqtgraph` (*class in optimeed.visualize.gui.widgets.graphsVisualWidget.smallGui*), 71
- ## H
- `hide()` (*TraceVisual method*), 72
- `hide_axes()` (*GraphVisual method*), 68
- `hide_points()` (*TraceVisual method*), 72
- `HowToPlotGraph` (*class in optimeed.core*), 43
- `HowToPlotGraph` (*class in optimeed.core.linkDataGraph*), 31
- `HowToPlotGraph` (*class in optimeed.visualize*), 105
- `hvRecursive()` (*HyperVolume method*), 50
- `HyperVolume` (*class in optimeed.optimize.optiAlgorithms.convergence.hypervolume*), 50

## I

`indentParagraph()` (*in module optimeed consolidate*), 20

`indentParagraph()` (*in module optimeed.core*), 36, 39

indentParagraph() (in module *optimeed.core.tools*), 35  
 index() (in module *optimeed.core.ansi2html.style*), 23  
 index2() (in module *optimeed.core.ansi2html.style*), 23  
 initialize() (*ContextHandler* method), 74  
 initialize() (*MyTerminationCondition* method), 53  
 initialize\_output\_collection() (*Parametric\_analysis* method), 20, 21  
 initializeGL() (*widget\_openGL* method), 79, 82, 92, 101  
 Integer\_OptimizationVariable (class in *optimeed.optimize*), 58  
 Integer\_OptimizationVariable (class in *optimeed.optimize.optiVariable*), 55  
 integrate() (in module *optimeed.core*), 46  
 integrate() (in module *optimeed.core.additional\_tools*), 25  
 intensify() (in module *optimeed.core.ansi2html.style*), 23  
 InterfaceCharacterization (class in *optimeed.optimize*), 57  
 InterfaceCharacterization (class in *optimeed.optimize.characterization*), 47  
 InterfaceCharacterization (class in *optimeed.optimize.characterization.interfaceCharacterization*), 47  
 InterfaceConvergence (class in *optimeed.optimize.optiAlgorithms.convergence*), 52  
 InterfaceConvergence (class in *optimeed.optimize.optiAlgorithms.convergence.interfaceConvergence*), 51  
 InterfaceDevice (class in *optimeed.core*), 43  
 InterfaceDevice (class in *optimeed.core.interfaceDevice*), 31  
 InterfaceMathsToPhysics (class in *optimeed.optimize*), 57  
 InterfaceMathsToPhysics (class in *optimeed.optimize.mathsToPhysics*), 48  
 InterfaceMathsToPhysics (class in *optimeed.optimize.mathsToPhysics.interfaceMathsToPhysics*), 47  
 InterfaceObjCons (class in *optimeed.optimize*), 57  
 InterfaceObjCons (class in *optimeed.optimize.objAndCons*), 49  
 InterfaceObjCons (class in *optimeed.optimize.objAndCons.interfaceObjCons*), 49  
 interpolate() (*fast\_LUT\_interpolation* method), 24, 45  
 interpolate\_table() (in module *optimeed.core*), 45  
 interpolate\_table() (in module *optimeed.core.additional\_tools*), 24  
 InTriangle() (in module *optimeed.visualize.gui.widgets.openGLWidget.TriangulatePolygon*), 76  
 is\_empty() (*DataAnimationVisuals* method), 66, 84, 88, 90, 95, 99  
 is\_monobj (*EvolutionaryConvergence* attribute), 49, 51  
 is\_object\_selected() (in module *optimeed.visualize.gui.gui\_data\_selector*), 88  
 is\_scattered() (*Data* method), 29, 41, 112  
 is\_slave() (*LinkDataGraph* method), 32, 44, 105  
 is\_slave() (*LinkDataGraph.collection\_linker* method), 32, 44, 105  
 IsClockwise() (in module *optimeed.visualize.gui.widgets.openGLWidget.TriangulatePolygon*), 76  
 IsConvex() (in module *optimeed.visualize.gui.widgets.openGLWidget.TriangulatePolygon*), 76  
 isNonePrintMessage() (in module *optimeed.core*), 39  
 isNonePrintMessage() (in module *optimeed.core.tools*), 35  
 isOnWindows (in module *optimeed.visualize.gui.widgets.graphsVisualWidget.pyqtgraphRedefin*), 69  
**J**  
 json\_to\_obj() (in module *optimeed.core*), 35  
 json\_to\_obj() (in module *optimeed.core.myjson*), 33  
 json\_to\_obj\_safe() (in module *optimeed.core*), 36  
 json\_to\_obj\_safe() (in module *optimeed.core.myjson*), 33  
**K**  
 keyboard\_push\_action() (*DeviceDrawerInterface* method), 74, 86, 93, 102  
 keyboardPushAction() (*ContextHandler* method), 74  
 keyboardReleaseAction() (*ContextHandler* method), 74  
 keyPressEvent() (*gui\_mainWindow* method), 89, 98, 106, 110  
 keyPressEvent() (*widget\_graphs\_visual* method), 77, 80, 91, 100, 107, 109  
 keyPressEvent() (*widget\_openGL* method), 79, 82, 93, 102  
 KWARGS\_OPTIHISTO (*Optimizer* attribute), 56, 59  
**L**  
 launch\_optimization() (*OptimizationDisplayer* method), 97, 108  
 level() (in module *optimeed.core.ansi2html.style*), 23



LineItem (class in optimeed.visualize.gui.widgets.graphsVisualWidget.exampleActionOnClick.on\_click\_measure module optimeed.visualize.gui.widgets.openGLWidget.ContextHandler), 62  
 link\_axes() (widget\_graphs\_visual method), 77, 80, 91, 100, 107, 109  
 link\_collection\_to\_graph\_collection() (LinkDataGraph method), 32, 44, 105  
 LinkDataGraph (class in optimeed.core), 44  
 LinkDataGraph (class in optimeed.core.linkDataGraph), 31  
 LinkDataGraph (class in optimeed.visualize), 104  
 LinkDataGraph.\_collection\_linker (class in optimeed.core), 44  
 LinkDataGraph.\_collection\_linker (class in optimeed.core.linkDataGraph), 31  
 LinkDataGraph.\_collection\_linker (class in optimeed.visualize), 105  
 linkify() (in module optimeed.core.ansi2html.converter), 22  
 linkXToGraph() (GraphVisual method), 69  
 linspace() (in module optimeed.core), 46  
 linspace() (in module optimeed.core.additional\_tools), 24  
 ListDataStruct (class in optimeed.core), 37  
 ListDataStruct (class in optimeed.core.collection), 26  
 log\_mode() (GraphVisual method), 69  
**M**  
 main() (in module optimeed.core.ansi2html.converter), 23  
 manage\_list() (in module optimeed.visualize.gui.gui\_data\_selector), 88  
 map\_index() (DataAnimationTrace method), 87  
 map\_vt100\_box\_code() (in module optimeed.core.ansi2html.converter), 22  
 MaterialRenderingProperties (class in optimeed.visualize.gui.widgets.openGLWidget.Materials\_visualize), 75  
 MathsToPhysics (class in optimeed.optimize), 57  
 MathsToPhysics (class in optimeed.optimize.mathsToPhysics), 48  
 MathsToPhysics (class in optimeed.optimize.mathsToPhysics.mathsToPhysics), 48  
 merge() (Graphs method), 31, 43, 113  
 merge() (ListDataStruct method), 27, 38  
 meshPolygon() (in module optimeed.visualize.gui.widgets.openGLWidget.TriangularMeshPolygon), 76  
 minimumSizeHint() (widget\_openGL method), 79, 82, 92, 101  
 MODE\_LIGHT (in module optimeed.visualize.gui.widgets.openGLWidget.ContextHandler), 70  
 MODE\_ZOOM (in module optimeed.visualize.gui.widgets.openGLWidget.ContextHandler), 73  
 modify\_paintElems() (TraceVisual.\_ModifiedPaintElem method), 71  
 MODULE\_TAG (in module optimeed.core.myjson), 33  
 mouseClicAction() (ContextHandler method), 74  
 mouseMotionAction() (ContextHandler method), 74  
 mouseMoveEvent() (widget\_openGL method), 79, 82, 93, 102  
 mousePressEvent() (widget\_openGL method), 79, 82, 93, 102  
 mouseWheelAction() (ContextHandler method), 74  
 MultiList (class in optimeed.optimize.optiAlgorithms.convergence.hypervolume), 50  
 MultiList.Node (class in optimeed.optimize.optiAlgorithms.convergence.hypervolume), 50  
 MultiObjective\_GA (class in optimeed.optimize), 58  
 MultiObjective\_GA (class in optimeed.optimize.optiAlgorithms), 54  
 MultiObjective\_GA (class in optimeed.optimize.optiAlgorithms.multiObjective\_GA), 53  
 my\_fft() (in module optimeed.core), 46  
 my\_fft() (in module optimeed.core.additional\_tools), 24  
 my\_fourier() (in module optimeed.core), 46  
 my\_fourier() (in module optimeed.core.additional\_tools), 25  
 myAxis (class in optimeed.visualize.gui.widgets.graphsVisualWidget.pyqtgraphReDefinition), 71  
 MyConvergence (class in optimeed.optimize.optiAlgorithms.multiObjective\_GA), 52  
 MyGenerator (class in optimeed.optimize.optiAlgorithms.multiObjective\_GA), 53  
 myGraphicsLayout (class in optimeed.visualize.gui.widgets.graphsVisualWidget.pyqtgraphReDefinition), 70  
 myGraphicsLayoutWidget (class in optimeed.visualize.gui.widgets.graphsVisualWidget.pyqtgraphReDefinition), 69  
 myItemSample (class in optimeed.visualize.gui.widgets.graphsVisualWidget.pyqtgraphReDefinition), 70

myLabelItem (class in opti- on\_click() (widget\_graphs\_visual method), 77, 80,  
                   meed.visualize.gui.widgets.graphsVisualWidget.pyqtgraph.Redefined), 106, 109  
 71  
 myLegend (class in opti- on\_click\_change\_symbol (class in opti-  
                   meed.visualize.gui.widgets.graphsVisualWidget.pyqtgraph.Redefined), 103  
 70  
 MyMapEvaluator (class in opti- on\_click\_change\_symbol (class in opti-  
                   meed.optimize.optiAlgorithms.multiObjective\_GA), meed.visualize.gui.widgets), 83  
 53  
 MyMultiprocessEvaluator (class in opti- on\_click\_change\_symbol (class in opti-  
                   meed.optimize.optiAlgorithms.multiObjective\_GA), meed.visualize.gui.widgets.graphsVisualWidget.examplesActionO  
 53  
 MyProblem (class in opti- on\_click\_change\_symbol (class in opti-  
                   meed.optimize.optiAlgorithms.multiObjective\_GA), meed.visualize.gui.widgets.graphsVisualWidget.examplesActionO  
 53  
 MyTerminationCondition (class in opti- on\_click\_copy\_something (class in opti-  
                   meed.optimize.optiAlgorithms.multiObjective\_GA), meed.visualize), 103  
 53  
 MyText (class in opti- on\_click\_copy\_something (class in opti-  
                   meed.visualize.gui.widgets.openGLWidget.ContextHandler), meed.visualize.gui.widgets), 83  
 74  
 myWindows (in module optimeed.visualize), 114  
 myWindows (in module optimeed.visualize.fastPlot), 98  
 65  
 on\_click\_copy\_something (class in opti-  
                   meed.visualize.gui.widgets.graphsVisualWidget.examplesActionO  
 61  
 on\_click\_extract\_pareto (class in opti-  
                   meed.visualize), 102  
 on\_click\_extract\_pareto (class in opti-  
                   meed.visualize.gui), 93  
 on\_click\_extract\_pareto (class in opti-  
                   meed.visualize.gui.widgets), 82  
 on\_click\_extract\_pareto (class in opti-  
                   meed.visualize.gui.widgets.graphsVisualWidget.examplesActionO  
 64  
 on\_click\_extract\_pareto (class in opti-  
                   meed.visualize.gui.widgets.graphsVisualWidget.examplesActionO  
 62  
 on\_click\_measure (class in optimeed.visualize), 103  
 on\_click\_measure (class in optimeed.visualize.gui),  
 94  
 on\_click\_measure (class in opti-  
                   meed.visualize.gui.widgets), 83  
 on\_click\_measure (class in opti-  
                   meed.visualize.gui.widgets.graphsVisualWidget.examplesActionO  
 67  
 on\_click\_measure (class in opti-  
                   meed.visualize.gui.widgets.graphsVisualWidget.examplesActionO  
 62  
 on\_graph\_click\_delete (class in opti-  
                   meed.visualize), 102  
 on\_graph\_click\_delete (class in opti-  
                   meed.visualize.gui), 93  
 on\_graph\_click\_delete (class in opti-

**N**  
 new\_figure() (WindowHolders method), 97, 114  
 new\_plot() (in module optimeed.visualize), 114  
 new\_plot() (in module optimeed.visualize.fastPlot),  
 98  
 new\_plot() (PlotHolders method), 97, 113  
 new\_plot() (WindowHolders method), 97, 114  
 next\_frame() (DataAnimationVisuals method), 66,  
 84, 87, 90, 95, 99  
 NLOpt\_Algorithm (class in opti-  
                   meed.optimize.optiAlgorithms.NLOpt\_Algorithm),  
 52  
 normalize() (in module opti-  
                   meed.visualize.gui.widgets.openGLWidget.quaternions),  
 76  
 NUMBER\_OF\_CORES (MultiObjective\_GA attribute),  
 53, 54, 58  
 NUMBER\_OF\_CORES (Parametric\_analysis attribute),  
 19, 21  
 NUMBER\_OF\_MODES (in module opti-  
                   meed.visualize.gui.widgets.openGLWidget.ContextHandler),  
 73

**O**  
 obj\_to\_json() (in module optimeed.core), 35  
 obj\_to\_json() (in module optimeed.core.myjson), 33  
 objectives (OptiHistoric.\_pointData attribute), 55  
 objectives\_per\_step (EvolutionaryConvergence  
                   attribute), 49, 51

`meed.visualize.gui.widgets)`, 82  
`on_graph_click_delete` (class in `optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick`), 103, 107  
64  
`on_graph_click_delete` (class in `optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick.on_click_delete`), 82  
61  
`on_graph_click_export` (class in `optimeed.visualize`), 102  
`on_graph_click_export` (class in `optimeed.visualize.gui`), 93  
`on_graph_click_export` (class in `optimeed.visualize.gui.widgets`), 82  
`on_graph_click_export` (class in `optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick`), 103, 107  
64  
`on_graph_click_export` (class in `optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick.on_click_export`), 83  
62  
`on_graph_click_interface` (class in `optimeed.visualize`), 104  
`on_graph_click_interface` (class in `optimeed.visualize.gui`), 95  
`on_graph_click_interface` (class in `optimeed.visualize.gui.widgets`), 84  
`on_graph_click_interface` (class in `optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick`), 103, 107  
65  
`on_graph_click_interface` (class in `optimeed.visualize.gui.widgets.widget_graphs_visual`), 77  
`on_graph_click_remove_trace` (class in `optimeed.visualize`), 103  
`on_graph_click_remove_trace` (class in `optimeed.visualize.gui`), 94  
`on_graph_click_remove_trace` (class in `optimeed.visualize.gui.widgets`), 83  
`on_graph_click_remove_trace` (class in `optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick`), 103, 107  
65  
`on_graph_click_remove_trace` (class in `optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick.on_click_remove_trace`), 82  
63  
`on_graph_click_showAnim` (class in `optimeed.visualize`), 104  
`on_graph_click_showAnim` (class in `optimeed.visualize.gui`), 96  
`on_graph_click_showAnim` (class in `optimeed.visualize.gui.widgets`), 85  
`on_graph_click_showAnim` (class in `optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick`), 103, 107  
67  
`on_graph_click_showAnim` (class in `optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick.on_click_showAnim`), 82  
61

optimeed.core.options (module), 33  
 optimeed.core.tools (module), 34  
 optimeed.optimize (module), 46  
 optimeed.optimize.characterization (module), 47  
 optimeed.optimize.characterization.characterization (module), 47  
 optimeed.optimize.characterization.interfaceCharacterization (module), 47  
 optimeed.optimize.mathsToPhysics (module), 47  
 optimeed.optimize.mathsToPhysics.interfaceMathsToPhysics (module), 47  
 optimeed.optimize.mathsToPhysics.mathsToPhysics (module), 48  
 optimeed.optimize.objAndCons (module), 48  
 optimeed.optimize.objAndCons.fastObjCons (module), 48  
 optimeed.optimize.objAndCons.interfaceObjCons (module), 48  
 optimeed.optimize.optiAlgorithms (module), 49  
 optimeed.optimize.optiAlgorithms.algorithmInitialization (module), 52  
 optimeed.optimize.optiAlgorithms.convergence (module), 49  
 optimeed.optimize.optiAlgorithms.convergenceAnalyticConvergence (module), 49  
 optimeed.optimize.optiAlgorithms.convergenceHyperVolume (module), 50  
 optimeed.optimize.optiAlgorithms.convergenceInterfaceConvergence (module), 51  
 optimeed.optimize.optiAlgorithms.multiObjective (module), 52  
 optimeed.optimize.optiAlgorithms.NLOpt\_Algorithm (module), 52  
 optimeed.optimize.optimizer (module), 55  
 optimeed.optimize.optiVariable (module), 54  
 optimeed.visualize (module), 60  
 optimeed.visualize.displayOptimization (module), 96  
 optimeed.visualize.fastPlot (module), 97  
 optimeed.visualize.gui (module), 60  
 optimeed.visualize.gui.gui\_collection\_experiment (module), 86  
 optimeed.visualize.gui.gui\_data\_animation (module), 86  
 optimeed.visualize.gui.gui\_data\_selector (module), 88  
 optimeed.visualize.gui.gui\_mainWindow (module), 89  
 optimeed.visualize.gui.widgets (module), 60  
 optimeed.visualize.gui.widgets.graphsVisualWidget (module), 60  
 optimeed.visualize.gui.widgets.graphsVisualWidget (module), 60  
 optimeed.visualize.gui.widgets.graphsVisualWidget (module), 61  
 optimeed.visualize.gui.widgets.graphsVisualWidget (module), 61  
 optimeed.visualize.gui.widgets.graphsVisualWidget (module), 62  
 optimeed.visualize.gui.widgets.graphsVisualWidget (module), 62  
 optimeed.visualize.gui.widgets.graphsVisualWidget (module), 62  
 optimeed.visualize.gui.widgets.graphsVisualWidget (module), 63  
 optimeed.visualize.gui.widgets.graphsVisualWidget (module), 63  
 optimeed.visualize.gui.widgets.graphsVisualWidget (module), 67  
 optimeed.visualize.gui.widgets.graphsVisualWidget (module), 69  
 optimeed.visualize.gui.widgets.graphsVisualWidget (module), 71  
 optimeed.visualize.gui.widgets.graphsVisualWidget (module), 71  
 optimeed.visualize.gui.widgets.graphsVisualWidget (module), 73  
 optimeed.visualize.gui.widgets.graphsVisualWidget (module), 73  
 optimeed.visualize.gui.widgets.graphsVisualWidget (module), 74  
 optimeed.visualize.gui.widgets.graphsVisualWidget (module), 74  
 optimeed.visualize.gui.widgets.graphsVisualWidget (module), 75  
 optimeed.visualize.gui.widgets.graphsVisualWidget (module), 76  
 optimeed.visualize.gui.widgets.graphsVisualWidget (module), 76  
 optimeed.visualize.gui.widgets.widget\_graphs\_visualization (module), 77  
 optimeed.visualize.gui.widgets.widget\_line\_drawer (module), 78  
 optimeed.visualize.gui.widgets.widget\_menuButton (module), 79  
 optimeed.visualize.gui.widgets.widget\_openGL (module), 79  
 optimeed.visualize.gui.widgets.widget\_text (module), 79



OptimizationDisplayer (class in *optimeed.visualize*), 108  
 OptimizationDisplayer (class in *optimeed.visualize.displayOptimization*), 96  
 OptimizationVariable (class in *optimeed.optimize.optiVariable*), 54  
 Optimizer (class in *optimeed.optimize*), 59  
 Optimizer (class in *optimeed.optimize.optimizer*), 56  
 Option\_class (class in *optimeed consolidate*), 20  
 Option\_class (class in *optimeed.core*), 45  
 Option\_class (class in *optimeed.core.options*), 34  
 Option\_class (class in *optimeed.visualize*), 106  
 Options (class in *optimeed.core*), 45  
 Options (class in *optimeed.core.options*), 33

## P

paint() (*LineItem* method), 62  
 paint() (*myItemSample* method), 70  
 paint() (*myLegend* method), 70  
 paintEvent() (*widget\_line\_drawer* method), 66, 78, 81, 85, 92, 101  
 paintGL() (*widget\_openGL* method), 79, 82, 93, 102  
 Parametric\_analysis (class in *optimeed consolidate*), 21  
 Parametric\_analysis (class in *optimeed consolidate.parametric\_analysis*), 19  
 Parametric\_Collection (class in *optimeed consolidate*), 20  
 Parametric\_Collection (class in *optimeed consolidate.parametric\_analysis*), 19  
 Parametric\_minmax (class in *optimeed consolidate*), 21  
 Parametric\_minmax (class in *optimeed consolidate.parametric\_analysis*), 19  
 Parametric\_parameter (class in *optimeed consolidate*), 20  
 Parametric\_parameter (class in *optimeed consolidate.parametric\_analysis*), 19  
 partition() (in module *optimeed.core*), 46  
 partition() (in module *optimeed.core.additional\_tools*), 24  
 pause\_play() (*DataAnimationVisuals* method), 65, 84, 87, 90, 95, 99  
 PipeOptimization (class in *optimeed.optimize.optimizer*), 55  
 plot() (in module *optimeed.visualize*), 114  
 plot() (in module *optimeed.visualize.fastPlot*), 98  
 PlotHolders (class in *optimeed.visualize*), 113  
 PlotHolders (class in *optimeed.visualize.fastPlot*), 97  
 pol2cart() (in module *optimeed.core*), 46

pol2cart() (in module *optimeed.core.additional\_tools*), 24  
 POPULATION\_SIZE (*NLOpt\_Algorithm* attribute), 52  
 prepare() (*Ansi2HTMLConverter* method), 23, 24  
 preProcess() (*HyperVolume* method), 50  
 printIfShown() (in module *optimeed.core*), 36, 39  
 printIfShown() (in module *optimeed.core.tools*), 35  
 produce\_headers() (*Ansi2HTMLConverter* method), 23, 24  
 PURPLE (*text\_format* attribute), 20, 34, 38, 45

## Q

q\_conjugate() (in module *optimeed.visualize.gui.widgets.openGLWidget.quaternions*), 76  
 q\_mult() (in module *optimeed.visualize.gui.widgets.openGLWidget.quaternions*), 76  
 q\_to\_axisangle() (in module *optimeed.visualize.gui.widgets.openGLWidget.quaternions*), 76  
 q\_to\_mat4() (in module *optimeed.visualize.gui.widgets.openGLWidget.quaternions*), 76  
 quicksort() (in module *optimeed.core*), 46  
 quicksort() (in module *optimeed.core.additional\_tools*), 24  
 qv\_mult() (in module *optimeed.visualize.gui.widgets.openGLWidget.quaternions*), 76

## R

read\_to\_unicode() (in module *optimeed.core.ansi2html.util*), 23  
 Real\_OptimizationVariable (class in *optimeed.optimize*), 58  
 Real\_OptimizationVariable (class in *optimeed.optimize.optiVariable*), 54  
 reconstitute\_signal() (in module *optimeed.core*), 46  
 reconstitute\_signal() (in module *optimeed.core.additional\_tools*), 24  
 RED (*text\_format* attribute), 20, 34, 38, 45  
 Red\_material (in module *optimeed.visualize.gui.widgets.openGLWidget.Materials\_visual*), 75  
 redraw() (*ContextHandler* method), 74  
 reformatXYtoList() (in module *optimeed.visualize.gui.widgets.openGLWidget.TriangulatePolygon*), 76  
 refreshTraceList() (*guiPyqtgraph* method), 71, 86, 93, 102, 108, 113  
 reinsert() (*MultiList* method), 51  
 remove() (*MultiList* method), 51

remove\_element\_from\_graph() (*LinkData-Graph method*), 32, 44, 105  
 remove\_elements\_from\_trace() (*LinkData-Graph method*), 32, 44, 105  
 remove\_feature() (*GraphVisual method*), 68  
 remove\_graph() (*Graphs method*), 30, 43, 113  
 remove\_trace() (*Graph method*), 29, 42  
 remove\_trace() (*Graphs method*), 30, 42, 112  
 remove\_trace() (*LinkDataGraph method*), 32, 44, 105  
 Repr\_lines (class in *optimeed.visualize*), 103  
 Repr\_lines (class in *optimeed.visualize.gui*), 94  
 Repr\_lines (class in *optimeed.visualize.gui.widgets*), 83  
 Repr\_lines (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick*), 64  
 Repr\_lines (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick.click.showinfo*), 63  
 Repr\_opengl (class in *optimeed.visualize*), 103  
 Repr\_opengl (class in *optimeed.visualize.gui*), 94  
 Repr\_opengl (class in *optimeed.visualize.gui.widgets*), 83  
 Repr\_opengl (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick*), 64  
 Repr\_opengl (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick.click.showinfo*), 63  
 reset() (*\_State method*), 22  
 reset() (*AlgorithmInterface method*), 52  
 reset() (*Graphs method*), 31, 43, 113  
 reset() (*gui\_collection\_exporter method*), 86, 90, 99  
 reset() (*on\_graph\_click\_delete method*), 62, 64, 82, 93, 102  
 reset() (*PlotHolders method*), 97, 113  
 reset() (*TraceVisual.\_ModifiedPaintElem method*), 72  
 reset\_all() (*DataAnimationVisuals method*), 65, 84, 87, 90, 95, 99  
 reset\_all\_brushes() (*TraceVisual method*), 73  
 reset\_all\_symbolPens() (*TraceVisual method*), 73  
 reset\_brush() (*TraceVisual method*), 73  
 reset\_distance() (*on\_click\_measure method*), 63, 67, 84, 95, 104  
 reset\_graph() (*on\_graph\_click\_export method*), 62, 64, 82, 93, 102  
 reset\_paintElem() (*TraceVisual.\_ModifiedPaintElem method*), 72  
 reset\_symbol() (*TraceVisual method*), 73  
 reset\_symbolPen() (*TraceVisual method*), 73  
 resizeEvent() (*myAxis method*), 71  
 resizeGL() (*widget\_openGL method*), 79, 82, 93, 102  
 resizeWindowAction() (*ContextHandler method*), 74  
 rgetattr() (in module *optimeed consolidate*), 20  
 rgetattr() (in module *optimeed.core*), 36, 39  
 rgetattr() (in module *optimeed.core.tools*), 35  
 rsetattr() (in module *optimeed consolidate*), 20  
 rsetattr() (in module *optimeed.core*), 39  
 rsetattr() (in module *optimeed.core.tools*), 35  
 Rule (class in *optimeed.core.ansi2html.style*), 23  
 run() (*DataAnimationVisuals method*), 66, 84, 88, 90, 95, 99  
 run() (*gui\_mainWindow method*), 89, 98, 106, 110  
 run() (*GuiDataSelector method*), 88  
 run() (*Parametric\_analysis method*), 19, 21  
 run\_optimization() (*Optimizer method*), 56, 59

## S

save() (*AutosaveStruct method*), 26, 37  
 save() (*ListDataStruct method*), 26, 37  
 save() (*OptiHistoric method*), 56  
 SaveableObject (class in *optimeed.core*), 36  
 SaveableObject (class in *optimeed.core.myjson*), 33  
 SCHEME (in module *optimeed.core.ansi2html.style*), 23  
 scrollable\_widget\_text (class in *optimeed.visualize.gui.widgets.widget\_text*), 64  
 set\_actionOnClick() (*widget\_graphs\_visual method*), 78, 81, 91, 100, 107, 109  
 set\_actionOnClick() (*gui\_mainWindow method*), 89, 98, 106, 110  
 set\_actionsOnClick() (*OptimizationDisplayer method*), 97, 108  
 set\_all\_options() (*Option\_class method*), 20, 34, 45, 106  
 set\_article\_template() (*widget\_graphs\_visual method*), 78, 81, 92, 101, 107, 110  
 set\_attribute\_data() (*ListDataStruct method*), 26, 37  
 set\_attribute\_equation() (*ListDataStruct method*), 26, 37  
 set\_attribute\_selectors() (*Container\_attribute\_selector method*), 88  
 set\_brush() (*TraceVisual method*), 72  
 set\_brushes() (*TraceVisual method*), 73  
 set\_collection() (*gui\_collection\_exporter method*), 86, 90, 99  
 set\_color() (*TraceVisual method*), 72  
 set\_color\_palette() (*GraphVisual method*), 68  
 set\_convergence() (*OptiHistoric method*), 56  
 set\_curr\_brush() (*DataAnimationTrace method*), 87  
 set\_currFigure() (*WindowHolders method*), 97, 114  
 set\_data() (*Data method*), 27, 40, 110

- set\_data() (*ListDataStruct* method), 26, 37  
 set\_data\_at\_index() (*ListDataStruct* method), 26, 37  
 set\_device() (*PipeOptimization* method), 55  
 set\_deviceDrawer() (*ContextHandler* method), 74  
 set\_deviceDrawer() (*widget\_openGL* method), 79, 82, 92, 101  
 set\_deviceToDraw() (*ContextHandler* method), 74  
 set\_deviceToDraw() (*widget\_openGL* method), 79, 82, 92, 101  
 set\_evaluationFunction() (*MultiObjective\_GA* method), 53, 54, 58  
 set\_evaluationFunction() (*NLOpt\_Algorithm* method), 52  
 set\_filename() (*AutosaveStruct* method), 25, 37  
 set\_font() (*myLegend* method), 70  
 set\_fontLabel() (*GraphVisual* method), 68  
 set\_fontLegend() (*GraphVisual* method), 68  
 set\_fontTicks() (*GraphVisual* method), 67  
 set\_graph\_disposition() (*myGraphicsLayout* method), 70  
 set\_graph\_disposition() (*widget\_graphs\_visual* method), 77, 80, 91, 100, 106, 109  
 set\_graph\_properties() (*GraphVisual* method), 68  
 set\_historic() (*PipeOptimization* method), 55  
 set\_idle\_brush() (*DataAnimationTrace* method), 87  
 set\_indices\_points\_to\_plot() (*Data* method), 29, 41, 112  
 set\_info() (*DataStruct\_Interface* method), 25, 36  
 set\_info() (*gui\_collection\_exporter* method), 86, 90, 99  
 set\_info() (*OptiHistoric* method), 56  
 set\_label\_pos() (*GraphVisual* method), 68  
 set\_label\_pos() (*myAxis* method), 71  
 set\_legend() (*GraphVisual* method), 68  
 set\_lims() (*GraphVisual* method), 68  
 set\_lines() (*widget\_line\_drawer* method), 66, 78, 81, 85, 92, 101  
 set\_max\_opti\_time() (*Optimizer* method), 56, 59  
 set\_maxtime() (*MultiObjective\_GA* method), 53, 54, 58  
 set\_maxtime() (*NLOpt\_Algorithm* method), 52  
 set\_number\_ticks() (*myAxis* method), 71  
 set\_numberTicks() (*GraphVisual* method), 67  
 set\_offset() (*myItemSample* method), 70  
 set\_offset\_sample() (*myLegend* method), 70  
 set\_optimizer() (*Optimizer* method), 56, 59  
 set\_option() (*Options* method), 34, 45  
 set\_optionValue() (*Option\_class* method), 20, 34, 45, 106  
 set\_permutations() (*Data* method), 29, 41, 111  
 set\_points\_at\_step() (*EvolutionaryConvergence* method), 49, 51  
 set\_pop\_size() (*ConvergenceManager* method), 52  
 set\_position() (*myLegend* method), 71  
 set\_refreshTime() (*DataAnimationVisuals* method), 66, 84, 88, 90, 95, 99  
 set\_results() (*OptiHistoric* method), 56  
 set\_same\_master() (*LinkData-Graph\_collection\_linker* method), 32, 44, 105  
 set\_self() (*Options* method), 34, 45  
 set\_space\_sample\_label() (*myLegend* method), 70  
 set\_specialButtonsMapping() (*ContextHandler* method), 74  
 set\_symbol() (*TraceVisual* method), 73  
 set\_symbolPen() (*TraceVisual* method), 73  
 set\_symbolPens() (*TraceVisual* method), 73  
 set\_text() (*scrollable\_widget\_text* method), 80  
 set\_text() (*widget\_text* method), 66, 79, 82, 84, 93, 102  
 set\_title() (*GraphVisual* method), 68  
 set\_title() (*in module optimeed.visualize*), 114  
 set\_title() (*in module optimeed.visualize.fastPlot*), 98  
 set\_title() (*PlotHolders* method), 97, 113  
 set\_title() (*widget\_graphs\_visual* method), 78, 81, 92, 101, 107, 110  
 set\_title() (*WindowHolders* method), 97, 114  
 set\_width\_cell() (*myItemSample* method), 70  
 set\_width\_cell\_sample() (*myLegend* method), 70  
 setText() (*myLabelItem* method), 71  
 shouldTerminate() (*MyTerminationCondition* method), 53  
 show() (*in module optimeed.visualize*), 114  
 show() (*in module optimeed.visualize.fastPlot*), 98  
 show() (*TraceVisual* method), 72  
 show() (*WindowHolders* method), 97, 114  
 show\_all() (*DataAnimationTrace* method), 87  
 show\_all() (*DataAnimationVisuals* method), 65, 84, 87, 90, 95, 99  
 SHOW\_CONSTRAINTS (*OptimizationDisplayer* attribute), 96, 108  
 SHOW\_CURRENT (*in module optimeed.core*), 39, 43  
 SHOW\_CURRENT (*in module optimeed.core.commonImport*), 27  
 SHOW\_DEBUG (*in module optimeed.core*), 39, 43  
 SHOW\_DEBUG (*in module optimeed.core.commonImport*), 27  
 SHOW\_ERROR (*in module optimeed.core*), 39, 43  
 SHOW\_ERROR (*in module optimeed.core.commonImport*), 27  
 SHOW\_INFO (*in module optimeed.core*), 39, 43

- SHOW\_INFO (in module *optimeed.core.commonImport*), 27
- SHOW\_WARNING (in module *optimeed.core*), 36, 39, 40, 43
- SHOW\_WARNING (in module *optimeed.core.commonImport*), 27
- showEvent() (*widget\_menuButton* method), 79, 81, 92, 101
- signal\_graph\_changed (*widget\_graphs\_visual* attribute), 77, 80, 91, 100, 106, 109
- signal\_has\_exported (*gui\_collection\_exporter* attribute), 86, 89, 98
- signal\_has\_reset (*gui\_collection\_exporter* attribute), 86, 89, 98
- signal\_must\_update (*TraceVisual* attribute), 72
- signal\_must\_update (*widget\_graphs\_visual* attribute), 77, 80, 91, 100, 106, 109
- signal\_must\_update (*widget\_line\_drawer* attribute), 66, 78, 81, 85, 92, 101
- signal\_optimization\_over (*OptimizationDisplayer* attribute), 96, 108
- Silver\_material (in module *optimeed.visualize.gui.widgets.openGLWidget.Materials\_visual*), 75
- sizeHint() (*widget\_openGL* method), 79, 81, 92, 101
- slider\_handler() (*DataAnimationVisuals* method), 66, 84, 87, 90, 95, 99
- SLIDER\_MAXIMUM\_VALUE (*DataAnimationVisuals* attribute), 65, 84, 87, 90, 95, 99
- SLIDER\_MINIMUM\_VALUE (*DataAnimationVisuals* attribute), 65, 84, 87, 90, 95, 99
- software\_version() (in module *optimeed.core*), 38
- software\_version() (in module *optimeed.core.tools*), 34
- sortByDimension() (*HyperVolume* method), 50
- sparse\_subset() (in module *optimeed.core*), 46
- sparse\_subset() (in module *optimeed.core.additional\_tools*), 25
- SpecialButtonsMapping (class in *optimeed.visualize.gui.widgets.openGLWidget.ContextHandler*), 74
- start\_autorefresh() (*OptimizationDisplayer* method), 97, 108
- start\_autosave() (*AutosaveStruct* method), 26, 37
- start\_qt\_mainloop() (in module *optimeed.visualize*), 98, 110
- start\_qt\_mainloop() (in module *optimeed.visualize.gui*), 89
- start\_qt\_mainloop() (in module *optimeed.visualize.gui.gui\_mainWindow*), 89
- Steel\_material (in module *optimeed.visualize.gui.widgets.openGLWidget.Materials\_visual*), 75
- stop\_autorefresh() (*OptimizationDisplayer* method), 97, 108
- stop\_autosave() (*AutosaveStruct* method), 26, 37
- stop\_qt\_mainloop() (in module *optimeed.visualize*), 98, 110
- stop\_qt\_mainloop() (in module *optimeed.visualize.gui*), 89
- stop\_qt\_mainloop() (in module *optimeed.visualize.gui.gui\_mainWindow*), 89
- str\_all\_attr() (in module *optimeed.core*), 39
- str\_all\_attr() (in module *optimeed.core.tools*), 35
- symbol\_isfilled() (*Data* method), 28, 40, 110
- ## T
- text\_format (class in *optimeed consolidate*), 20
- text\_format (class in *optimeed.core*), 38, 45
- text\_format (class in *optimeed.core.tools*), 34
- theActionOnUpdate (*GuiDataSelector* attribute), 88
- time (*OptiHistoric.\_pointData* attribute), 55
- to\_css\_classes() (*\_State* method), 22
- toggle() (*TraceVisual* method), 72
- TraceVisual (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.traceVisual*), 71
- TraceVisual.\_ModifiedPaintElem (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.traceVisual*), 71
- truncate() (in module *optimeed.core*), 39
- truncate() (in module *optimeed.core.tools*), 35
- ## U
- UNDERLINE (*text\_format* attribute), 20, 34, 38, 45
- universalPath() (in module *optimeed.core*), 39
- universalPath() (in module *optimeed.core.tools*), 35
- update() (*GraphVisual* method), 69
- update\_graphs() (*LinkDataGraph* method), 32, 44, 105
- update\_graphs() (*widget\_graphs\_visual* method), 77, 80, 91, 100, 106, 109
- update\_widget\_w\_animation() (*DataAnimationLines* method), 60, 67, 85, 96, 104
- update\_widget\_w\_animation() (*DataAnimationOpenGL* method), 60, 66, 85, 96, 104
- update\_widget\_w\_animation() (*DataAnimationOpenGLwText* method), 60, 66, 85, 96, 104
- update\_widget\_w\_animation() (*DataAnimationVisualswText* method), 61, 67, 85, 96, 104
- updateChildren() (*Graphs* method), 30, 42, 112
- updateSize() (*myLegend* method), 70
- updateTrace() (*TraceVisual* method), 72
- useOpenGL() (*myGraphicsLayoutWidget* method), 70



## V

VERSION (in module *optimeed*), 114

VT100\_BOX\_CODES (in module *optimeed.core.ansi2html.converter*), 22

## W

wheelEvent() (*widget\_openGL* method), 79, 82, 93, 102

WHITE (*text\_format* attribute), 20, 34, 38, 45

widget\_graphs\_visual (class in *optimeed.visualize*), 99, 106, 108

widget\_graphs\_visual (class in *optimeed.visualize.gui*), 90

widget\_graphs\_visual (class in *optimeed.visualize.gui.widgets*), 80

widget\_graphs\_visual (class in *optimeed.visualize.gui.widgets.widget\_graphs\_visual*), 77

widget\_line\_drawer (class in *optimeed.visualize*), 101

widget\_line\_drawer (class in *optimeed.visualize.gui*), 92

widget\_line\_drawer (class in *optimeed.visualize.gui.widgets*), 81, 85

widget\_line\_drawer (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick*), 66

widget\_line\_drawer (class in *optimeed.visualize.gui.widgets.widget\_line\_drawer*), 78

widget\_menuButton (class in *optimeed.visualize*), 101

widget\_menuButton (class in *optimeed.visualize.gui*), 92

widget\_menuButton (class in *optimeed.visualize.gui.widgets*), 81

widget\_menuButton (class in *optimeed.visualize.gui.widgets.widget\_menuButton*), 79

widget\_openGL (class in *optimeed.visualize*), 101

widget\_openGL (class in *optimeed.visualize.gui*), 92

widget\_openGL (class in *optimeed.visualize.gui.widgets*), 81

widget\_openGL (class in *optimeed.visualize.gui.widgets.widget\_openGL*), 79

widget\_text (class in *optimeed.visualize*), 102

widget\_text (class in *optimeed.visualize.gui*), 93

widget\_text (class in *optimeed.visualize.gui.widgets*), 82, 84

widget\_text (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick*), 66

widget\_text (class in *optimeed.visualize.gui.widgets.widget\_text*), 79

WindowHolders (class in *optimeed.visualize*), 113

WindowHolders (class in *optimeed.visualize.fastPlot*), 97

## Y

YELLOW (*text\_format* attribute), 20, 34, 38, 45

Yellow\_Emerald\_material (in module *optimeed.visualize.gui.widgets.openGLWidget.Materials\_visual*), 75