
optimeed

Release 1.0

Jan 21, 2020

Contents

1	Requirements	3
2	Installation	5
3	Quickstart	7
3.1	Quickstart Optimization	7
3.2	Quickstart Visualization	9
3.3	Loading and saving data	12
4	Gallery	15
4.1	Gallery	15
5	License and support	17
5.1	License and Support	17
6	API	19
6.1	API optimeed	19
7	Developer guide	113
7.1	Developer documentation	113
	Python Module Index	115
	Index	117

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

CHAPTER 1

Requirements

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

CHAPTER 2

Installation

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

```
pip install optimeed
```

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

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


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

3.1 Quickstart Optimization

An optimization process can be presented as following:

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

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

```
"""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, theMachine):
        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, 4) # 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

6.1 API optimeed

6.1.1 Subpackages

`consolidate`

`parametric_analysis`

Module Contents

```

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

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

    get_reference_device (self)

    get_analyzed_attribute (self)

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

    get_values (self)

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

    NUMBER_OF_CORES = 1

    run (self)
        Instantiates input arguments for analysis

```

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

Package Contents

```
class Option_class
```

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

```
getPath_workspace ()
```

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

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

Recursively get an attribute from object. Extends getattr method

Parameters

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

Returns

```
class text_format
```

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

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

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

Bases: optimeed.core.collection.Collection

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

Abstract class for a parametric parameter


```

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

```

core

Subpackages

ansi2html

converter

Module Contents

```

ANSI_FULL_RESET = 0
ANSI_INTENSITY_INCREASED = 1
ANSI_INTENSITY_REDUCED = 2
ANSI_INTENSITY_NORMAL = 22
ANSI_STYLE_ITALIC = 3
ANSI_STYLE_NORMAL = 23
ANSI_BLINK_SLOW = 5
ANSI_BLINK_FAST = 6
ANSI_BLINK_OFF = 25
ANSI_UNDERLINE_ON = 4
ANSI_UNDERLINE_OFF = 24
ANSI_CROSSED_OUT_ON = 9
ANSI_CROSSED_OUT_OFF = 29
ANSI_VISIBILITY_ON = 28
ANSI_VISIBILITY_OFF = 8

```

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

_latex_template = \documentclass{scrartcl}
usepackage[utf8]{inputenc}      usepackage{fancyvrb}      usepackage[usenames,dvipsnames]{xcolor}      %%
definecolor{red-sd}{HTML}{7ed2d2}
title{%(title)s}
fvset{commandchars=\{\}}
begin{document}
begin{Verbatim} %(content)s end{Verbatim} end{document}

_html_template
class _State
    Bases: object
    reset (self)
    adjust (self, ansi_code, parameter=None)
    to_css_classes (self)
linkify (line, latex_mode)
map_vt100_box_code (char)
_needs_extra_newline (text)
class CursorMoveUp
    Bases: object
class Ansi2HTMLConverter (latex=False, inline=False, dark_bg=True, line_wrap=True,
    font_size='normal', linkify=False, escaped=True, markup_lines=False,
    output_encoding='utf-8', scheme='ansi2html', title='')
    Bases: object
    Convert Ansi color codes to CSS+HTML
```

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

apply_regex (*self*, *ansi*)

_apply_regex (*self*, *ansi*, *styles_used*)

_collapse_cursor (*self*, *parts*)

Act on any CursorMoveUp commands by deleting preceding tokens

prepare (*self*, *ansi*="", *ensure_trailing_newline*=False)

Load the contents of 'ansi' into this object

attrs (*self*)

Prepare attributes for the template

convert (*self*, *ansi*, *full*=True, *ensure_trailing_newline*=False)

produce_headers (*self*)

main ()

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

`$ task burndown | ansi2html > burndown.html`

style

Module Contents

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

Bases: object

__str__ (*self*)

index (*r*, *g*, *b*)

color_component (*x*)

color (*r*, *g*, *b*)

level (*grey*)

index2 (*grey*)

SCHEME

intensify (*color*, *dark_bg*, *amount*=64)

get_styles (*dark_bg*=True, *line_wrap*=True, *scheme*='ansi2html')

util

Module Contents

read_to_unicode (*obj*)

Package Contents

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

Bases: `object`

Convert Ansi color codes to CSS+HTML

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

apply_regex (*self, ansi*)

_apply_regex (*self, ansi, styles_used*)

_collapse_cursor (*self, parts*)

Act on any CursorMoveUp commands by deleting preceding tokens

prepare (*self, ansi="", ensure_trailing_newline=False*)

Load the contents of 'ansi' into this object

attrs (*self*)

Prepare attributes for the template

convert (*self, ansi, full=True, ensure_trailing_newline=False*)

produce_headers (*self*)

collection

Module Contents

```
class DataStruct_Interface
```

get_info (*self*)

Get simple string describing the datastructure

set_info (*self, info*)

Set simple string describing the datastructure

__str__ (*self*)

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

Structure that provides automated save of DataStructures

__str__ (*self*)

get_filename (*self*)

Get set filename

set_filename (*self, filename, change_filename_if_exists*)

Parameters

- **filename** – Filename to set
- **change_filename_if_exists** – If already exists, create a new filename

stop_autosave (*self*)

Stop autosave

```

start_autosave (self, timer_autosave)
    Start autosave

save (self, safe_save=True)
    Save

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

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

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

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

    get_data (self)
        Get full list of datas

    set_data (self, theData)
        Set full list of datas

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

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

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

        Parameters

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

Returns
        list

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

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

        Returns list

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

        Parameters indices – list of indices to delete

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

        Parameters

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

```

merge (*self*, *collection*)

Merge a collection with the current collection

Parameters **collection** – Collection to merge

color_palette

Module Contents

default_palette (*N*)

blackOnly (*N*)

dark2 (*N*)

commonImport

Module Contents

SHOW_WARNING = 0

SHOW_INFO = 1

SHOW_ERROR = 2

SHOW_DEBUG = 3

SHOW_CURRENT

graphs

Module Contents

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

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

set_data (*self*, *x: list*, *y: list*)

Overwrites current datapoints with new set

get_x (*self*)

Get x coordinates of datapoints

get_symbolsize (*self*)

Get size of the symbols

symbol_isfilled (*self*)

Check if symbols has to be filled or not

get_symbolOutline (*self*)

Get color factor of outline of symbols

get_length_data (*self*)

Get number of points

get_xlim(*self*)
Get x limits of viewbox

get_ylim(*self*)
Get y limits of viewbox

get_y(*self*)
Get y coordinates of datapoints

get_color(*self*)
Get color of the line

get_width(*self*)
Get width of the line

get_number_of_points(*self*)
Get number of points

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

Returns x (list), y (list)

get_permutations(*self*)
Return the transformation 'permutation': `xplot[i] = xdata[permutation[i]]`

get_invert_permutations(*self*)
Return the inverse of permutations: `xdata[i] = xplot[revert[i]]`

get_dataIndex_from_graphIndex(*self*, *index_graph_point*)
From an index given in graph, recovers the index of the data.

Parameters *index_graph_point* – Index in the graph
Returns index of the data

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

Parameters *index_graph_point_list* – List of Index in the graph
Returns List of index of the data

get_graphIndex_from_dataIndex(*self*, *index_data*)
From an index given in the data, recovers the index of the graph.

Parameters *index_data* – Index in the data
Returns index of the graph

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

Parameters *index_data_list* – List of Index in the data
Returns List of index of the graph

set_permutations(*self*, *permutations*)
Set permutations between datapoints of the trace

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

get_x_label (*self*)
Get x label of the trace

get_y_label (*self*)
Get y label of the trace

get_legend (*self*)
Get name of the trace

get_symbol (*self*)
Get symbol

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

delete_point (*self*, *index_point*)
Delete a point from the datapoints

is_scattered (*self*)
Delete a point from the datapoints

set_indices_points_to_plot (*self*, *indices*)
Set indices points to plot

get_indices_points_to_plot (*self*)
Get indices points to plot

get_linestyle (*self*)
Get linestyle

__str__ (*self*)

export_str (*self*)
Method to save the points constituting the trace

class Graph

Simple graph container that contains several traces

add_trace (*self*, *data*)
Add a trace to the graph

Parameters *data* – *Data*

Returns id of the created trace

remove_trace (*self*, *idTrace*)
Delete a trace from the graph

Parameters *idTrace* – id of the trace to delete

get_trace (*self*, *idTrace*)
Get data object of *idTrace*

Parameters *idTrace* – id of the trace to get

Returns *Data*

get_all_traces (*self*)
Get all the traces id of the graph

export_str (*self*)

class Graphs

Contains several *Graph*

updateChildren (*self*)

add_trace_firstGraph (*self*, *data*, *updateChildren=True*)

Same as add_trace, but only if graphs has only one id :param data: :param updateChildren: :return:

add_trace (*self*, *idGraph*, *data*, *updateChildren=True*)

Add a trace to the graph

Parameters

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

Returns id of the created trace

remove_trace (*self*, *idGraph*, *idTrace*, *updateChildren=True*)

Remove the trace from the graph

Parameters

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

get_first_graph (*self*)

Get id of the first graph

Returns id of the first graph

get_graph (*self*, *idGraph*)

Get graph object at idgraph

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

Returns *Graph*

get_all_graphs_ids (*self*)

Get all ids of the graphs

Returns list of id graphs

get_all_graphs (*self*)

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

add_graph (*self*, *updateChildren=True*)

Add a new graph

Returns id of the created graph

remove_graph (*self*, *idGraph*)

Delete a graph

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

add_update_method (*self*, *childObject*)

Add a callback each time a graph is modified.

Parameters **childObject** – method without arguments

export_str (*self*)

Export all the graphs in text

Returns str

merge (*self*, *otherGraphs*)

```
reset (self)
```

```
interfaceDevice
```

Module Contents

```
class InterfaceDevice
```

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

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

Copy the attribute values of machine_to_assign to self. The references are not lost.

Parameters

- **machine_to_assign** – InterfaceDevice
- **resetAttribute** –

```
linkDataGraph
```

Module Contents

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

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

Add id_col to exclude from the graph

```
__str__ (self)
```

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

```
get_collection (self)
```

```
get_kwargs (self)
```

```
get_id (self)
```

```
class LinkDataGraph
```

```
class _collection_linker
```

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

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

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

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

Parameters

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

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

```

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

```

myjson

Module Contents

MODULE_TAG = `__module__`

CLASS_TAG = `__class__`

EXCLUDED_TAGS

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*)

`obj_to_json` (*theObj*)

Extract the json dictionary from the object. The data saved are automatically detected, using typehints. ex: x: int=5 will be saved, x=5 won't.

`encode_str_json` (*theStr*)

`decode_str_json` (*theStr*)

`options`

Module Contents

`class Options`

`get_name` (*self, idOption*)

`get_value` (*self, idOption*)

`add_option` (*self, idOption, name, value*)

`set_option` (*self, idOption, value*)

`copy` (*self*)

`set_self` (*self, the_options*)

`__str__` (*self*)

`class Option_class`

`get_optionValue` (*self, optionId*)

`set_optionValue` (*self, optionId, value*)

`get_all_options` (*self*)

`set_all_options` (*self, options*)

```
add_option (self, idOption, name, value)
```

tools

Module Contents

```
class text_format
```

```

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

```

```
software_version ()
```

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

```
create_unique_dirname (dirname)
```

```
applyEquation (objectIn, s)
```

Apply literal expression based on an object

Parameters

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

Returns value (float)

```
arithmeticEval (s)
```

```
isNonePrintMessage (theObject, theMessage, show_type=SHOW_INFO)
```

```
getPath_workspace ()
```

```
getLineInfo (lvl=1)
```

```
printIfShown (theStr, show_type=SHOW_DEBUG, isToPrint=True, appendTypeName=True)
```

```
universalPath (thePath)
```

```
add_suffix_to_path (thePath, suffix)
```

```
get_object_attrs (obj)
```

```
cart2pol (x, y)
```

pol2cart (*rho, phi*)

partition (*array, begin, end*)

quicksort (*array*)

rsetattr (*obj, attr, val*)

rgetattr (*obj, attr*)

Recursively get an attribute from object. Extends getattr method

Parameters

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

Returns

indentParagraph (*text_in, indent_level=1*)

dist (*p, q*)

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

sparse_subset (*points, r*)

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

integrate (*x, y*)

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

Parameters

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

Returns integral value

my_fourier (*x, y, n, L*)

Fourier analys

Parameters

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

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

linspace (*start, stop, npoints*)

truncate (*theStr, truncsize*)

str_all_attr (*theObject, max_recursion_level*)

get_2D_pareto (*xList, yList, max_X=True, max_Y=True*)

get_ND_pareto (*objectives_list, are_maxobjectives_list=None*)

Return the N-D pareto front

Parameters

- **objectives_list** – list of list of objectives: example [[0,1], [1,1], [2,2]]

- **are_maxobjectives_list** – for each objective, tells if they are to be maximized or not: example [True, False]. Default: False

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

derivate (*t, y*)

class fast_LUT_interpolation (*independent_variables, dependent_variables*)

Class designed for fast interpolation in look-up table when successive searches are called often. Otherwise use griddata

interpolate (*self, point, fill_value=np.nan*)

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

delete_indices_from_list (*indices, theList*)

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

Package Contents

getPath_workspace ()

obj_to_json (*theObj*)

Extract the json dictionary from the object. The data saved are automatically detected, using typehints. ex: x: int=5 will be saved, x=5 won't.

json_to_obj (*json_dict*)

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

json_to_obj_safe (*json_dict, cls*)

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

encode_str_json (*theStr*)

decode_str_json (*theStr*)

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`

```
class ListDataStruct
```

Bases: `optimeed.core.collection.DataStruct_Interface`

```
    _INFO_STR = info
```

```
    _DATA_STR = data
```

```
    save (self, filename)
```

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

```
    add_data (self, data_in)
```

Add a data to the list

```
    get_data (self)
```

Get full list of datas

```
    set_data (self, theData)
```

Set full list of datas

set_data_at_index (*self, data_in, index*)

Replace data at specific index

set_attribute_data (*self, the_attribute, the_value*)

Set attribute to all data

set_attribute_equation (*self, attribute_name, equation_str*)

Advanced method to set the value of attribute_name from equation_str

Parameters

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

Returns

get_list_attributes (*self, attributeName*)

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

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

Returns list

delete_points_at_indices (*self, indices*)

Delete several elements from the Collection

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

export_xls (*self, excelFilename, excelsheet='Sheet1', mode='w'*)

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

Parameters

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

merge (*self, collection*)

Merge a collection with the current collection

Parameters **collection** – Collection to merge

class text_format

PURPLE = [95m

CYAN = [96m

DARKCYAN = [36m

BLUE = [94m

GREEN = [92m

YELLOW = [93m

WHITE = [30m

RED = [91m

BOLD = [1m

UNDERLINE = [4m

END = [0m

software_version ()

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

create_unique_dirname (*dirname*)

applyEquation (*objectIn, s*)

Apply literal expression based on an object

Parameters

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

Returns value (float)

arithmeticEval (*s*)

isNonePrintMessage (*theObject, theMessage, show_type=SHOW_INFO*)

getPath_workspace ()

getLineInfo (*lvl=1*)

printIfShown (*theStr, show_type=SHOW_DEBUG, isToPrint=True, appendTypeName=True*)

universalPath (*thePath*)

add_suffix_to_path (*thePath, suffix*)

get_object_attrs (*obj*)

cart2pol (*x, y*)

pol2cart (*rho, phi*)

partition (*array, begin, end*)

quicksort (*array*)

rsetattr (*obj, attr, val*)

rgetattr (*obj, attr*)

Recursively get an attribute from object. Extends getattr method

Parameters

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

Returns

indentParagraph (*text_in, indent_level=1*)

dist (*p, q*)

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

sparse_subset (*points, r*)

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

integrate (*x, y*)

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

Parameters

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

Returns integral value

my_fourier (*x, y, n, L*)

Fourier analys

Parameters

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

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

linspace (*start, stop, npoints*)

truncate (*theStr, truncsize*)

str_all_attr (*theObject, max_recursion_level*)

get_2D_pareto (*xList, yList, max_X=True, max_Y=True*)

get_ND_pareto (*objectives_list, are_maxobjectives_list=None*)

Return the N-D pareto front

Parameters

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

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

derivate (*t, y*)

class fast_LUT_interpolation (*independent_variables, dependent_variables*)

Class designed for fast interpolation in look-up table when successive searches are called often. Otherwise use griddata

interpolate (*self, point, fill_value=np.nan*)

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

delete_indices_from_list (*indices, theList*)

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

SHOW_WARNING = 0

SHOW_INFO = 1

SHOW_ERROR = 2

SHOW_DEBUG = 3

SHOW_CURRENT

printIfShown (*theStr, show_type=SHOW_DEBUG, isToPrint=True, appendTypeName=True*)

SHOW_WARNING = 0

```
class Data (x: list, y: list, x_label="", y_label="", legend="", is_scattered=False, transfo_x=lambda self-  
Data, x: x, transfo_y=lambda 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*, *machine_to_assign*, *resetAttribute=False*)
Copy the attribute values of *machine_to_assign* to *self*. The references are not lost.

Parameters

- **machine_to_assign** – InterfaceDevice
- **resetAttribute** –

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

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

__str__ (*self*)

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

```
get_collection (self)
get_kwargs (self)
get_id (self)
class LinkDataGraph

    class _collection_linker

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



```

    get_graph_and_trace_from_collection (self, idCollection)
        Reverse search: from a collection, get the associated graph

    get_mappingData_graph (self, idGraph)

    get_mappingData_trace (self, idGraph, idTrace)

class text_format

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

class Options

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

class Option_class

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

```

optimize

Subpackages

characterization

characterization

Module Contents

class Characterization

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

compute (*self, theDevice*)

interfaceCharacterization

Module Contents

class InterfaceCharacterization

Bases: *optimeed.core.options.Option_class*

Interface for the evaluation of a device

__str__ (*self*)

Package Contents

class InterfaceCharacterization

Bases: *optimeed.core.options.Option_class*

Interface for the evaluation of a device

__str__ (*self*)

class Characterization

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

compute (*self, theDevice*)

mathsToPhysics

interfaceMathsToPhysics

Module Contents

class InterfaceMathsToPhysics

Bases: *optimeed.core.options.Option_class*

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

mathsToPhysics

Module Contents

class MathsToPhysics

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

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

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

fromPhysToMaths (*self, theDevice, theOptimizationVariables*)

__str__ (*self*)

Package Contents

class MathsToPhysics

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

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

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

fromPhysToMaths (*self, theDevice, theOptimizationVariables*)

__str__ (*self*)

class InterfaceMathsToPhysics

Bases: *optimeed.core.options.Option_class*

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

objAndCons

fastObjCons

Module Contents

class FastObjCons (*constraintEquation, name=None*)

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

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

compute (*self, theDevice*)

get_name (*self*)

interfaceObjCons

Module Contents

class InterfaceObjCons

Bases: *optimeed.core.options.Option_class*

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

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, theOptimization-
        Variables, theOptimizationAlgorithm=default['Algo'], theCharacteriza-
        tion=default['Charac'], theMathsToPhysics=default['M2P'])
        Prepare the optimizer for the optimization.
        Parameters
            • theDevice – object of type InterfaceDevice
            • theCharacterization – object of type InterfaceCharacterization
            • theMathsToPhysics – object of type InterfaceMathsToPhysics
            • theObjectiveList – list of objects of type InterfaceObjCons
            • theConstraintList – list of objects of type InterfaceObjCons
            • theOptimizationAlgorithm – list of objects of type AlgorithmInterface
            • theOptimizationVariables – list of objects of type OptimizationVariable
        Returns PipeOptimization
    run_optimization (self)
        Perform the optimization.
        Returns Collection of the best optimized machines
    set_max_opti_time (self, max_time_sec)

```

evaluateObjectiveAndConstraints (*self*, *x*)

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

This function is NOT process safe: “self.” is actually a FORK in multiprocessing algorithms. It means that the motor originally contained in *self*. is modified only in the fork, and only gathered by reaching the end of the fork. It is not (yet?) possible to access this motor on the main process before the end of the fork. This behaviour could be changed by using pipes or Managers.

Parameters *x* – Input mathematical vector from optimization algorithm

Returns dictionary, containing objective values (list of scalar), constraint values (list of scalar), and other info (motor, time)

callback_on_evaluation (*self*, *returnedValues*)

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

formatInfo (*self*)

Package Contents

class InterfaceCharacterization

Bases: *optimeed.core.options.Option_class*

Interface for the evaluation of a device

__str__ (*self*)

class Characterization

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

compute (*self*, *theDevice*)

class MathsToPhysics

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

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

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

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

__str__ (*self*)

class InterfaceMathsToPhysics

Bases: *optimeed.core.options.Option_class*

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

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

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

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

compute (*self*, *theDevice*)

get_name (*self*)

class InterfaceObjConsBases: *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 machines

```
set_max_opti_time (self, max_time_sec)
```

```
evaluateObjectiveAndConstraints (self, x)
```

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

This function is NOT process safe: “self.” is actually a FORK in multiprocessing algorithms. It means that the motor originally contained in self. is modified only in the fork, and only gathered by reaching the end of the fork. It is not (yet?) possible to access this motor on the main process before the end of the fork. This behaviour could be changed by using pipes or Managers.

Parameters **x** – Input mathematical vector from optimization algorithm

Returns dictionary, containing objective values (list of scalar), constraint values (list of scalar), and other info (motor, time)

```
callback_on_evaluation (self, returnedValues)
```

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

```
formatInfo (self)
```

visualize

Subpackages

gui

Subpackages

widgets

Subpackages

graphsVisualWidget

Subpackages

examplesActionOnClick

on_click_anim

Module Contents

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

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

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

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

    Same as DataAnimationLines but also with text

    update_widget_w_animation (self, key, index, the_data_animation)

class on_graph_click_showAnim (theLinkDataGraph, theAnimation)
    Bases:                                     optimeed.visualize.gui.widgets.widget_graphs_visual.
                                                on_graph_click_interface

    On click: add or remove an element to animate

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

    get_name (self)
```

on_click_change_symbol

Module Contents

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

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

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

    get_name (self)
```

on_click_copy_something

Module Contents

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

    On Click: copy something

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

    get_name (self)
```

on_click_delete

Module Contents

```
class delete_gui
    Bases: PyQt5.QtWidgets.QMainWindow

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


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

apply (*self*)

reset (*self*)

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

get_name (*self*)

on_click_export_collection

Module Contents

class on_graph_click_export (*theDataLink*)

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

On click: export the selected points

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

reset_graph (*self*)

get_name (*self*)

on_click_extract_pareto

Module Contents

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

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

On click: extract the pareto from the cloud of points

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

get_name (*self*)

on_click_remove_trace

Module Contents

class on_graph_click_remove_trace (*theDataLink*)

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

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

get_name (*self*)

`on_click_showinfo`

Module Contents

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

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

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

class `DataInformationVisuals`

delete_visual (*self*, *theVisual*)

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

get_new_index (*self*)

curr_index (*self*)

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

Action to perform when a point in the graph has been clicked: Creates new window displaying the device and its informations

get_name (*self*)

class `Repr_lines` (*attribute_lines*)

get_widget (*self*, *theNewDevice*)

class `Repr_opengl` (*DeviceDrawer*)

get_widget (*self*, *theNewDevice*)

Package Contents

class `on_graph_click_delete` (*theDataLink*)

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

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

apply (*self*)

reset (*self*)

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

get_name (*self*)

class `on_graph_click_export` (*theDataLink*)

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

On click: export the selected points

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

reset_graph (*self*)

```

    get_name (self)

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

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

    get_name (self)

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

class DataInformationVisuals

    delete_visual (self, theVisual)

    add_visual (self, theVisual, theTrace, indexPoint)

    get_new_index (self)

    curr_index (self)

    graph_clicked (self, theGraphVisual, index_graph, index_trace, indices_points)
        Action to perform when a point in the graph has been clicked: Creates new window displaying the device
        and its informations

    get_name (self)

class Repr_opengl (DeviceDrawer)

    get_widget (self, theNewDevice)

class Repr_lines (attribute_lines)

    get_widget (self, theNewDevice)

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

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

    get_name (self)

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

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

    get_name (self)

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

```

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

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

get_name (*self*)

class on_graph_click_interface

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

class DataAnimationVisuals (*id=0*, *window_title='Animation'*)

Bases: `PyQt5.QtWidgets.QMainWindow`

Spawns a gui that includes button to create animations nicely when paired with *widget_graphs_visual*

SLIDER_MAXIMUM_VALUE = 500

SLIDER_MINIMUM_VALUE = 1

add_trace (*self*, *trace_id*, *element_list*, *theTrace*)

Add a trace to the animation.

Parameters

- **trace_id** – id of the trace
- **element_list** – List of elements to save: `[[OpenGL_item1, text_item1], [OpenGL_item2, text_item2], ... [OpenGL_itemN, text_itemN]]`
- **theTrace** – `TraceVisual`

Returns

add_elementToTrace (*self*, *trace_id*, *indexPoint*)

delete_point (*self*, *trace_id*, *thePoint*)

reset_all (*self*)

delete_all (*self*)

pause_play (*self*)

show_all (*self*)

next_frame (*self*)

slider_handler (*self*)

frame_selector (*self*)

set_refreshTime (*self*)

is_empty (*self*)

run (*self*)

closeEvent (*self*, *_*)

contains_trace (*self*, *trace_id*)

export_picture (*self*)

class widget_text (*theText*, *is_light=False*, *convertToHtml=False*)

Bases: `PyQt5.QtWidgets.QLabel`

Widget able to display a text

set_text (*self*, *theText*, *convertToHtml=False*)

Set the text to display

```

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

    Widget allowing to display several lines easily

    signal_must_update

    on_update_signal (self, listOfLines)

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

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

    paintEvent (self, event, painter=None)

    get_extrema_lines (self)

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

    Implements DataAnimationVisuals to show opengl drawing

    update_widget_w_animation (self, key, index, the_data_animation)

    export_widget (self, painter)

    delete_key_widgets (self, key)

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

    Implements DataAnimationVisuals to show opengl drawing and text

    update_widget_w_animation (self, key, index, the_data_animation)

    get_interesting_elements (self, devices_list)

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

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

    export_widget (self, painter)

    delete_key_widgets (self, key)

    update_widget_w_animation (self, key, index, the_data_animation)

    get_interesting_elements (self, devices_list)

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

    Same as DataAnimationLines but also with text

    update_widget_w_animation (self, key, index, the_data_animation)

class on_graph_click_showAnim (theLinkDataGraph, theAnimation)
    Bases: optimeed.visualize.gui.widgets.widget_graphs_visual.
    on_graph_click_interface

    On click: add or remove an element to animate

```

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

graphVisual

Module Contents

```
class GraphVisual (theWidgetGraphVisual)
```

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

```
set_fontTicks (self, fontSize, fontname=None)  
Set font of the ticks
```

Parameters

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

```
set_numberTicks (self, number, axis)  
Set the number of ticks to be displayed
```

Parameters

- **number** – Number of ticks for the axis
- **axis** – Axis (string, “bottom”, “left”, “right”, “top”)

Returns

```
set_fontLabel (self, fontSize, color='#000', fontname=None)  
Set font of the axis labels
```

Parameters

- **fontSize** – font size
- **color** – color in hexadecimal (str)
- **fontname** – name of the font

```
get_legend (self)  
Get the legend
```

```
get_axis (self, axis)  
Get the axis
```

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

Returns axis object

```
set_fontLegend (self, font_size, font_color, fontname=None)
```

```
set_label_pos (self, orientation, x_offset=0, y_offset=0)
```

```
set_color_palette (self, palette)
```

```
apply_palette (self)
```

```
hide_axes (self)
```

```
add_feature (self, theFeature)  
To add any pyqtgraph item to the graph
```

remove_feature (*self*, *theFeature*)

To remove any pyqtgraph item from the graph

add_data (*self*, *idGraph*, *theColor*, *theData*)

set_graph_properties (*self*, *theTrace*)

This function is automatically called on creation of the graph

set_lims (*self*, *xlim*, *ylim*)

Set limits of the graphs, xlim or ylim = [val_low, val_high]. Or None.

add_trace (*self*, *idTrace*, *theTrace*)

Add a TraceVisual to the graph, with index idTrace

set_legend (*self*)

Set default legend options (color and font)

set_title (*self*, *titleName*, ***kwargs*)

Set title of the graph

Parameters titleName – title to set

get_trace (*self*, *idTrace*)

Return the TraceVisual correspondong to the index idTrace

get_all_traces (*self*)

Return a dictionary {idtrace: TraceVisual}.

delete_trace (*self*, *idTrace*)

Delete the trace of index idTrace

delete (*self*)

Delete the graph

linkXToGraph (*self*, *graph*)

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

update (*self*)

Update the traces contained in the graph

fast_update (*self*)

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

axis_equal (*self*)

grid_off (*self*)

Turn off grid

pyqtgraphRedefine

Module Contents

isOnWindows

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

class myGraphicsLayoutWidget (*parent=None*, ***kwargs*)

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

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

Overwrited to fix bad antialiasing while using openGL

class myGraphicsLayout

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

addItem (*self*, *item*, *row=None*, *col=None*, *rowspan=1*, *colspan=1*)

Add an item to the layout and place it in the next available cell (or in the cell specified). The item must be an instance of a QGraphicsWidget subclass.

set_graph_disposition (*self*, *item*, *row=1*, *col=1*, *rowspan=1*, *colspan=1*)

Function to modify the position of an item in the list

Parameters

- **item** – WidgetPlotItem to set
- **row** – Row
- **col** – Column
- **rowspan** –
- **colspan** –

Returns

class myItemSample (*item*)

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

set_offset (*self*, *offset*)

set_width_cell (*self*, *width*)

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

Overwrites to make matlab-like samples

class myLegend (*size=None*, *offset=(30, 30)*, *is_light=False*)

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

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

set_space_sample_label (*self*, *theSpace*)

To set the gap between the sample and the label

set_offset_sample (*self*, *offset*)

To tune the offset between the sample and the text

set_width_cell_sample (*self*, *width*)

Set width of sample

updateSize (*self*)

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

Overwrites to flush left

apply_width_sample (*self*)

set_font (*self*, *font_size*, *font_color*, *fontname=None*)

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

Overwrited to select background color

set_position (*self, position, offset*)
Set the position of the legend, in a corner.

Parameters

- **position** – String (NW, NE, SW, SE), indicates which corner the legend is close
- **offset** – Tuple (xoff, yoff), x and y offset from the edge

Returns

class myLabelItem

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

setText (*self, text, **args*)
Overwrited to add font-family to options

class myAxis (*orientation*)

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

get_label_pos (*self*)
Overwrited to place label closer to the axis

resizeEvent (*self, ev=None*)
Overwrited to place label closer to the axis

set_label_pos (*self, orientation, x_offset=0, y_offset=0*)

set_number_ticks (*self, number*)

smallGui

Module Contents

class guiPyqtgraph (*graphsVisual, **kwargs*)

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

refreshTraceList (*self*)
Refresh all the traces

traceVisual

Module Contents

class TraceVisual (*theColor, theData, theWGPlot, highlight_last*)

Bases: `PyQt5.QtCore.QObject`

Defines a trace in a graph.

class _ModifiedPaintElem

Hidden class to manage brushes or pens

add_modified_paintElem (*self, index, newPaintElem*)

modify_paintElems (*self, paintElemsIn_List*)

Apply transformation to paintElemsIn_List

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

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

reset_paintElem (*self*, *index*)
Remove transformation of point index

reset (*self*)

signal_must_update

hide_points (*self*)
Hide all the points

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

set_color (*self*, *color*)
Set colour of the trace, argument as tuple (r,g,b)

get_base_symbol_brush (*self*)
Get symbol brush configured for this trace, return `pg.QBrush`

get_base_pen (*self*)
Get pen configured for this trace, return `pg.QPen`

get_base_symbol_pen (*self*)
Get symbol pen configured for this trace, return `pg.QPen`

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

get_symbol (*self*, *size*)
Get actual symbols for the trace. If the symbols have been modified: return a list which maps each points to a symbol. Otherwise: return :meth:TraceVisual.get_base_symbol()

updateTrace (*self*)
Forces the trace to refresh.

get_length (*self*)
Return number of data to plot

hide (*self*)
Hides the trace

show (*self*)
Shows the trace

toggle (*self*, *boolean*)
Toggle the trace (hide/show)

get_data (*self*)
Get data to plot `Data`

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

set_brush (*self*, *indexPoint*, *newbrush*, *update=True*)
Set the symbol brush for a specific point:

Parameters

- **indexPoint** – Index of the point (in the graph) to modify
- **newbrush** – either `QBrush` or tuple (r, g, b) of the new brush
- **update** – if True, update the trace afterwards. This is slow operation.

set_symbol (*self*, *indexPoint*, *newSymbol*, *update=True*)

Set the symbol shape for a specific point:

Parameters

- **indexPoint** – Index of the point (in the graph) to modify
- **newSymbol** – string of the new symbol (e.g.: 'o')
- **update** – if True, update the trace afterwards. This is slow operation.

set_brushes (*self*, *list_indexPoint*, *list_newbrush*)

Same as [set_brush\(\)](#) but by taking a list as input

reset_brush (*self*, *indexPoint*, *update=True*)

Reset the brush of the point indexpoint

reset_all_brushes (*self*)

Reset all the brushes

reset_symbol (*self*, *indexPoint*, *update=True*)

Reset the symbol shape of the point indexpoint

get_symbolPens (*self*, *size*)

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

set_symbolPen (*self*, *indexPoint*, *newPen*, *update=True*)

Set the symbol shape for a specific point:

Parameters

- **indexPoint** – Index of the point (in the graph) to modify
- **newPen** – QPen item or tuple of the color (r,g,b)
- **update** – if True, update the trace afterwards. This is slow operation.

set_symbolPens (*self*, *list_indexPoint*, *list_newpens*)

Same as [set_symbolPen\(\)](#) but by taking a list as input

reset_symbolPen (*self*, *indexPoint*)

Reset the symbol pen of the point indexpoint

reset_all_symbolPens (*self*)

Reset all the symbol pens

OpenGLWidget

ContextHandler

Module Contents

MODE_ZOOM = 0

MODE_ROTATION = 1

MODE_LIGHT = 2

NUMBER_OF_MODES = 3

CLIC_LEFT = 0

```
CLIC_RIGHT = 1
class SpecialButtonsMapping
class MyText (color, fontSize, theStr, windowPosition)
class ContextHandler

    set_specialButtonsMapping (self, theSpecialButtonsMapping)
    set_deviceDrawer (self, theDeviceDrawer)
    set_deviceToDraw (self, theDeviceToDraw)
    resizeWindowAction (self, new_width, new_height)
    mouseWheelAction (self, deltaAngle)
    mouseClicAction (self, button, my_x, y)
    mouseMotionAction (self, my_x, y)
    keyboardPushAction (self, key)
    keyboardReleaseAction (self, key, my_x, y)
    __draw_axis__ (self)
    redraw (self)
    get_text_to_write (self)
    __lightingInit__ (self)
    initialize (self)
    __reset__ (self)
```

DeviceDrawerInterface

Module Contents

```
class DeviceDrawerInterface

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

Materials_visual

Module Contents

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

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

```

__dif3__ = [0, 0, 0, 0]
__amb3__ = [0, 0, 0, 0]
__shin__ = 0
getSpec3 (self)
getDif3 (self)
getAmb3 (self)
getShin (self)
activateMaterialProperties (self, alpha=1)

```

Emerald_material

Yellow_Emerald_material

Brass_material

Bronze_material

Silver_material

Steel_material

Copper_material

Chrome_material

Blue_material

Red_material

OpenGLFunctions_Library

Module Contents

```

draw_closedPolygon (xClockWise, yClockWise)
draw_extrudeZ (xList, yList, zExtrude)
draw_triList (theTriList)
draw_lines (x, z)
draw_spiralSheet (innerRadius, thickness, length, theAngle, n, reverseDirection=False)
draw_spiralFront (innerRadius, thicknessMaterial, thicknessSpiral, z0, theAngle, n, reverseDirection=False)
draw_spiralFull (innerRadius, outerRadius, thicknessMaterial, thicknessSpiral, length, n)
draw_spiral (innerRadius, outerRadius, thicknessMaterial, thicknessSpiral, length, cutAngle, n)
draw_simple_rectangle (width, height)
draw_rectangle (rIn, length, thickness, angle, reverseDirection=False)
draw_2Dring (innerRadius, outerRadius, z0, theAngle, n, reverseDirection=False)
draw_2Dring_diff_angle (innerRadius, outerRadius, angle_in, angle_out, n, reverseDirection=False)
draw_tubeSheet (radius, length, theAngle, n, reverseDirection=False)
draw_cylinder (innerRadius, outerRadius, length, n, translate=0)

```

draw_part_cylinder (*innerRadius, outerRadius, length, angle, n, translate=0, drawSides=True*)
draw_disk (*innerRadius, outerRadius, n, translate=0*)
draw_part_disk (*innerRadius, outerRadius, thickness, angle, n, translate=0*)
draw_part_disk_diff_angles (*innerRadius, outerRadius, thickness, angle_in, angle_out, n*)
draw_carved_disk (*innerRadius, outerRadius, carvedRin, carvedRout, thickness, depth, angle, n, translate=0*)
draw_part_cylinder_throat (*rIn, rOut, rOutThroat, length, lengthThroat, angle, n, translate=0*)
drawWireTube (*diameter, xa, ya, xb, yb, n=50, translateZ=0*)

TriangulatePolygon

Module Contents

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

quaternions

Module Contents

normalize (*v, tolerance=0.001*)
q_mult (*q1, q2*)
q_conjugate (*q*)
qv_mult (*q1, v1*)
axisangle_to_q (*v, theta*)
q_to_axisangle (*q*)
q_to_mat4 (*q*)

widget_graphs_visual

Module Contents

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

```
class widget_graphs_visual (theGraphs, **kwargs)
```

```
    Bases: PyQt5.QtWidgets.QWidget
```

Widget element to draw a graph. The traces and graphs to draw are defined in `Graphs` taken as argument. This widget is linked to the excellent third-party library `pyqtgraph`, under MIT license

```
signal_must_update
```

```
signal_graph_changed
```

```
set_graph_disposition (self, indexGraph, row=1, col=1, rowspan=1, colspan=1)
```

Change the graphs disposition.

Parameters

- **indexGraph** – index of the graph to change
- **row** – row where to place the graph
- **col** – column where to place the graph
- **rowspan** – number of rows across which the graph spans
- **colspan** – number of columns across which the graph spans

Returns

```
__create_graph (self, idGraph)
```

```
__check_graphs (self)
```

```
on_click (self, plotDataItem, clicked_points)
```

```
update_graphs (self, singleUpdate=True)
```

This method is used to update the graph. This is fast but NOT safe (especially when working with threads). To limit the risks, please use `self.signal_must_update.emit()` instead.

Parameters singleUpdate – if set to False, the graph will periodically refres each `self.refreshTime`

```
fast_update (self)
```

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

```
exportGraphs (self)
```

Export the graphs

```
link_axes (self)
```

```
get_graph (self, idGraph)
```

Get corresponding `GraphVisual` of the graph `idGraph`

```
keyPressEvent (self, event)
```

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

```
delete_graph (self, idGraph)
```

Delete the graph `idGraph`

```
delete (self)
```

```
get_all_graphsVisual (self)
```

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

```
get_layout_buttons (self)
```

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

```
set_actionOnClick (self, theActionOnClick)
```

Action to perform when the graph is clicked

Parameters **theActionOnClick** – *on_graph_click_interface*

Returns

set_title (*self*, *idGraph*, *titleName*, ***kwargs*)

Set title of the graph

Parameters

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

set_article_template (*self*, *graph_size_x*=8.8, *graph_size_y*=4.4, *legendPosition*='NW')

Method to set the graphs to article quality graph.

Parameters

- **graph_size_x** – width of the graph in cm
- **graph_size_y** – height of the graph in cm
- **legendPosition** – position of the legend (NE, SE, SW, NW)

Returns

widget_line_drawer

Module Contents

class widget_line_drawer (*minWinHeight*=300, *minWinWidth*=300, *is_light*=True)

Bases: `PyQt5.QtWidgets.QWidget`

Widget allowing to display several lines easily

signal_must_update

on_update_signal (*self*, *listOfLines*)

delete_lines (*self*, *key_id*)

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*)

widget_menuButton

Module Contents

class widget_menuButton (*theParentButton*)

Bases: `PyQt5.QtWidgets.QMenu`

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

showEvent (*self*, *QShowEvent*)

widget_openGL

Module Contents

```
class widget_openGL (parent=None)
    Bases: PyQt5.QtWidgets.QOpenGLWidget

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

    sizeHint (self)

    minimumSizeHint (self)

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

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

    initializeGL (self)

    paintGL (self)

    resizeGL (self, w, h)

    mousePressEvent (self, event)

    mouseMoveEvent (self, event)

    keyPressEvent (self, event)

    wheelEvent (self, QWheelEvent)
```

widget_text

Module Contents

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

    Widget able to display a text

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

class scrollable_widget_text (theText, is_light=False, convertToHtml=False)
    Bases: PyQt5.QtWidgets.QWidget

    Same as widget_text but scrollable

    set_text (self, theText, convertToHtml=False)
```

Package Contents

```
class widget_graphs_visual (theGraphs, **kwargs)
    Bases: PyQt5.QtWidgets.QWidget

    Widget element to draw a graph. The traces and graphs to draw are defined in Graphs taken as argument. This widget is linked to the excellent third-party library pyqtgraph, under MIT license
```

signal_must_update

signal_graph_changed

set_graph_disposition (*self*, *indexGraph*, *row=1*, *col=1*, *rowspan=1*, *colspan=1*)

Change the graphs disposition.

Parameters

- **indexGraph** – index of the graph to change
- **row** – row where to place the graph
- **col** – column where to place the graph
- **rowspan** – number of rows across which the graph spans
- **colspan** – number of columns across which the graph spans

Returns

__create_graph (*self*, *idGraph*)

__check_graphs (*self*)

on_click (*self*, *plotDataItem*, *clicked_points*)

update_graphs (*self*, *singleUpdate=True*)

This method is used to update the graph. This is fast but NOT safe (especially when working with threads). To limit the risks, please use `self.signal_must_update.emit()` instead.

Parameters singleUpdate – if set to False, the graph will periodically refres each `self.refreshTime`

fast_update (*self*)

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

exportGraphs (*self*)

Export the graphs

link_axes (*self*)

get_graph (*self*, *idGraph*)

Get corresponding `GraphVisual` of the graph `idGraph`

keyPressEvent (*self*, *event*)

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

delete_graph (*self*, *idGraph*)

Delete the graph `idGraph`

delete (*self*)

get_all_graphsVisual (*self*)

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

get_layout_buttons (*self*)

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

set_actionOnClick (*self*, *theActionOnClick*)

Action to perform when the graph is clicked

Parameters theActionOnClick – `on_graph_click_interface`

Returns

set_title (*self*, *idGraph*, *titleName*, ***kwargs*)

Set title of the graph

Parameters

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

set_article_template (*self*, *graph_size_x*=8.8, *graph_size_y*=4.4, *legendPosition*='NW')

Method to set the graphs to article quality graph.

Parameters

- **graph_size_x** – width of the graph in cm
- **graph_size_y** – height of the graph in cm
- **legendPosition** – position of the legend (NE, SE, SW, NW)

Returns

class widget_line_drawer (*minWinHeight*=300, *minWinWidth*=300, *is_light*=True)

Bases: `PyQt5.QtWidgets.QWidget`

Widget allowing to display several lines easily

signal_must_update

on_update_signal (*self*, *listOfLines*)

delete_lines (*self*, *key_id*)

Dele the lines :param key_id: id to delete :return:

set_lines (*self*, *listOfLines*, *key_id*=0, *pen*=None)

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

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

get_extrema_lines (*self*)

class widget_menuButton (*theParentButton*)

Bases: `PyQt5.QtWidgets.QMenu`

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

showEvent (*self*, *QShowEvent*)

class widget_openGL (*parent*=None)

Bases: `PyQt5.QtWidgets.QOpenGLWidget`

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

sizeHint (*self*)

minimumSizeHint (*self*)

set_deviceDrawer (*self*, *theDeviceDrawer*)

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

set_deviceToDraw (*self*, *theDeviceToDraw*)

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

initializeGL (*self*)

paintGL (*self*)

```
resizeGL (self, w, h)  
mousePressEvent (self, event)  
mouseMoveEvent (self, event)  
keyPressEvent (self, event)  
wheelEvent (self, QWheelEvent)  
  
class widget_text (theText, is_light=False, convertToHtml=False)  
    Bases: PyQt5.QtWidgets.QLabel  
    Widget able to display a text  
  
    set_text (self, theText, convertToHtml=False)  
        Set the text to display  
  
class on_graph_click_delete (theDataLink)  
    Bases: optimeed.visualize.gui.widgets.widget_graphs_visual.on_graph_click_interface  
    On Click: Delete the points from the graph, and save the modified collection  
  
    apply (self)  
  
    reset (self)  
  
    graph_clicked (self, theGraphVisual, index_graph, index_trace, indices_points)  
  
    get_name (self)  
  
class on_graph_click_export (theDataLink)  
    Bases: optimeed.visualize.gui.widgets.widget_graphs_visual.on_graph_click_interface  
    On click: export the selected points  
  
    graph_clicked (self, theGraphVisual, index_graph, index_trace, indices_points)  
  
    reset_graph (self)  
  
    get_name (self)  
  
class on_click_extract_pareto (theDataLink, max_x=False, max_y=False)  
    Bases: optimeed.visualize.gui.widgets.widget_graphs_visual.on_graph_click_interface  
    On click: extract the pareto from the cloud of points  
  
    graph_clicked (self, the_graph_visual, index_graph, index_trace, _)  
  
    get_name (self)  
  
class on_graph_click_showInfo (theLinkDataGraph, visuals=None)  
    Bases: optimeed.visualize.gui.widgets.widget_graphs_visual.on_graph_click_interface  
    On click: show informations about the points (loop through attributes)  
  
class DataInformationVisuals  
  
    delete_visual (self, theVisual)  
  
    add_visual (self, theVisual, theTrace, indexPoint)  
  
    get_new_index (self)
```

```

    curr_index (self)

graph_clicked (self, theGraphVisual, index_graph, index_trace, indices_points)
    Action to perform when a point in the graph has been clicked: Creates new window displaying the device
    and its informations

get_name (self)

class Repr_opengl (DeviceDrawer)

    get_widget (self, theNewDevice)

class Repr_lines (attribute_lines)

    get_widget (self, theNewDevice)

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

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

    get_name (self)

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

    On Click: copy something

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

    get_name (self)

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

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

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

    get_name (self)

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

class DataAnimationVisuals (id=0, window_title='Animation')
    Bases: PyQt5.QtWidgets.QMainWindow

    Spawns a gui that includes button to create animations nicely when paired with widget_graphs_visual

    SLIDER_MAXIMUM_VALUE = 500

    SLIDER_MINIMUM_VALUE = 1

    add_trace (self, trace_id, element_list, theTrace)
        Add a trace to the animation.

        Parameters
        • trace_id – id of the trace
        • element_list – List of elements to save: [[OpenGL_item1, text_item1],
            [OpenGL_item2, text_item2], ... [OpenGL_itemN, text_itemN]]

```

- **theTrace** – TraceVisual

Returns**add_elementToTrace** (*self*, *trace_id*, *indexPoint*)**delete_point** (*self*, *trace_id*, *thePoint*)**reset_all** (*self*)**delete_all** (*self*)**pause_play** (*self*)**show_all** (*self*)**next_frame** (*self*)**slider_handler** (*self*)**frame_selector** (*self*)**set_refreshTime** (*self*)**is_empty** (*self*)**run** (*self*)**closeEvent** (*self*, *_*)**contains_trace** (*self*, *trace_id*)**export_picture** (*self*)**class widget_text** (*theText*, *is_light=False*, *convertToHtml=False*)Bases: `PyQt5.QtWidgets.QLabel`

Widget able to display a text

set_text (*self*, *theText*, *convertToHtml=False*)

Set the text to display

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

Widget allowing to display several lines easily

signal_must_update**on_update_signal** (*self*, *listOfLines*)**delete_lines** (*self*, *key_id*)Delete the lines :param *key_id*: id to delete :return:**set_lines** (*self*, *listOfLines*, *key_id=0*, *pen=None*)Set the lines to display :param *listOfLines*: list of [*x1*, *y1*, *z1*, *x2*, *y2*, *z2*] corresponding to lines :param *key_id*: id of the trace :param *pen*: pen used to draw the lines :return:**paintEvent** (*self*, *event*, *painter=None*)**get_extrema_lines** (*self*)**class DataAnimationOpenGL** (*theOpenGLWidget*, *theId=0*, *window_title='Animation'*)Bases: `optimeed.visualize.gui.gui_data_animation.DataAnimationVisuals`Implements `DataAnimationVisuals` to show opengl drawing**update_widget_w_animation** (*self*, *key*, *index*, *the_data_animation*)

```

    export_widget (self, painter)

    delete_key_widgets (self, key)

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

    Implements DataAnimationVisuals to show opengl drawing and text

    update_widget_w_animation (self, key, index, the_data_animation)

    get_interesting_elements (self, devices_list)

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

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

    export_widget (self, painter)

    delete_key_widgets (self, key)

    update_widget_w_animation (self, key, index, the_data_animation)

    get_interesting_elements (self, devices_list)

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

    Same as DataAnimationLines but also with text

    update_widget_w_animation (self, key, index, the_data_animation)

class on_graph_click_showAnim (theLinkDataGraph, theAnimation)
    Bases: optimeed.visualize.gui.widgets.widget_graphs_visual.
            on_graph_click_interface

    On click: add or remove an element to animate

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

    get_name (self)

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

    refreshTraceList (self)
        Refresh all the traces

class DeviceDrawerInterface

    keyboard_push_action (self, theKey)

    get_colour_scalebar (self)

    get_colour_background (self)

    get_opengl_options (self)

```

gui_collection_exporter

Module Contents

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

gui_data_animation

Module Contents

```
class DataAnimationTrace (elements_list, theTrace)
    Contains all the element to animate for a trace
    class element_animation (elements)
        get (self)
        get_element_animations (self, itemNumber, index_in_show)
            Get the element to show :param itemNumber: item number (0 if only one think to draw) :param index_in_show: index in the list :return: The element to draw
        show_all (self)
        delete_all (self)
        get_indices_to_show (self)
        add_element (self, indexPoint)
        add_index_to_show (self, index)
        _remove_index_from_show (self, index)
        set_curr_brush (self, index_in_show)
        set_idle_brush (self, index_in_show)
        get_number_of_elements (self)
        map_index (self, index_in_show)
```



```

    get_base_pen (self)

class DataAnimationVisuals (id=0, window_title='Animation')
    Bases: PyQt5.QtWidgets.QMainWindow

    Spawns a gui that includes button to create animations nicely when paired with widget_graphs_visual

    SLIDER_MAXIMUM_VALUE = 500
    SLIDER_MINIMUM_VALUE = 1

    add_trace (self, trace_id, element_list, theTrace)
        Add a trace to the animation.

        Parameters
        • trace_id – id of the trace
        • element_list – List of elements to save: [[OpenGL_item1, text_item1],
            [OpenGL_item2, text_item2], ... [OpenGL_itemN, text_itemN]]
        • theTrace – TraceVisual

        Returns

    add_elementToTrace (self, trace_id, indexPoint)
    delete_point (self, trace_id, thePoint)
    reset_all (self)
    delete_all (self)
    pause_play (self)
    show_all (self)
    next_frame (self)
    slider_handler (self)
    frame_selector (self)
    set_refreshTime (self)
    is_empty (self)
    run (self)
    closeEvent (self, _)
    contains_trace (self, trace_id)
    export_picture (self)

```

```
gui_data_selector
```

Module Contents

```

app

class Action_on_selector_update

class Attribute_selector (attribute_name, value)

    add_child (self, child)

```

```
    get_children (self)
    get_name (self)
    get_min_max_attributes (self)
    __str__ (self)
class Container_attribute_selector (containerName)

    add_child (self, child)
    add_attribute_selector (self, attribute_selector)
    set_attribute_selectors (self, attribute_selectors)
    get_name (self)
    get_children (self)
    get_attribute_selectors (self)
    __str__ (self)
class GuiDataSelector (collections_in: CollectionsToVisualise, actionOnUpdate: ActionOnSelectorUpdate)
    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 (QWidgetList, 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)

```

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)  
    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 widget_menuButton (theParentButton)  
    Bases: PyQt5.QtWidgets.QMenu  
  
    Same as QMenu, but integrates it behind a button more easily.  
  
    showEvent (self, QShowEvent)  
class widget_opengl (parent=None)  
    Bases: PyQt5.QtWidgets.QOpenGLWidget  
  
    Interface that provides opengl capabilities. Ensures zoom, light, rotation, etc.  
  
    sizeHint (self)  
    minimumSizeHint (self)  
    set_deviceDrawer (self, theDeviceDrawer)  
        Set a drawer optimeed.visualize.gui.widgets.openglWidget.DeviceDrawerInterface.DeviceDrawerInterface  
    set_deviceToDraw (self, theDeviceToDraw)  
        Set the device to draw optimeed.InterfaceDevice.InterfaceDevice  
    initializeGL (self)  
    paintGL (self)  
    resizeGL (self, w, h)  
    mousePressEvent (self, event)  
    mouseMoveEvent (self, event)  
    keyPressEvent (self, event)  
    wheelEvent (self, QWheelEvent)  
class widget_text (theText, is_light=False, convertToHtml=False)  
    Bases: PyQt5.QtWidgets.QLabel  
  
    Widget able to display a text  
  
    set_text (self, theText, convertToHtml=False)  
        Set the text to display  
class guiPyqtgraph (graphsVisual, **kwargs)  
    Create a gui for pyqtgraph with trace selection options, export and action on clic choices  
  
    refreshTraceList (self)  
        Refresh all the traces
```

```
class DeviceDrawerInterface
```

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

```
    get_colour_scalebar (self)
```

```
    get_colour_background (self)
```

```
    get_opengl_options (self)
```

```
class on_graph_click_delete (theDataLink)
```

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

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

```
    apply (self)
```

```
    reset (self)
```

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

```
    get_name (self)
```

```
class on_graph_click_export (theDataLink)
```

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

On click: export the selected points

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

```
    reset_graph (self)
```

```
    get_name (self)
```

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

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

On click: extract the pareto from the cloud of points

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

```
    get_name (self)
```

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

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

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

```
class DataInformationVisuals
```

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

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

```
    get_new_index (self)
```

```
    curr_index (self)
```

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

Action to perform when a point in the graph has been clicked: Creates new window displaying the device and its informations

```
    get_name (self)
class Repr_opengl (DeviceDrawer)

    get_widget (self, theNewDevice)
class Repr_lines (attribute_lines)

    get_widget (self, theNewDevice)
class on_graph_click_remove_trace (theDataLink)
    Bases:                                     optimeed.visualize.gui.widgets.widget_graphs_visual.
                                              on_graph_click_interface
    graph_clicked (self, theGraphVisual, index_graph, index_trace, _)
    get_name (self)
class on_click_copy_something (theDataLink, functionStrFromDevice)
    Bases:                                     optimeed.visualize.gui.widgets.widget_graphs_visual.
                                              on_graph_click_interface
    On Click: copy something
    graph_clicked (self, the_graph_visual, index_graph, index_trace, indices_points)
    get_name (self)
class on_click_change_symbol (theLinkDataGraph)
    Bases:                                     optimeed.visualize.gui.widgets.widget_graphs_visual.
                                              on_graph_click_interface
    On Click: Change the symbol of the point that is clicked
    graph_clicked (self, theGraphVisual, index_graph, index_trace, indices_points)
    get_name (self)
class on_graph_click_interface
    Interface class for the action to perform when a point is clicked
class DataAnimationVisuals (id=0, window_title='Animation')
    Bases: PyQt5.QtWidgets.QMainWindow
    Spawns a gui that includes button to create animations nicely when paired with widget_graphs_visual
    SLIDER_MAXIMUM_VALUE = 500
    SLIDER_MINIMUM_VALUE = 1
    add_trace (self, trace_id, element_list, theTrace)
        Add a trace to the animation.

        Parameters
            • trace_id – id of the trace
            • element_list – List of elements to save: [[OpenGL_item1, text_item1],
                [OpenGL_item2, text_item2], ... [OpenGL_itemN, text_itemN]]
            • theTrace – TraceVisual

        Returns
    add_elementToTrace (self, trace_id, indexPoint)
```



```

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

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

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

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

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

```

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

displayOptimization

Module Contents

```
class OptimizationDisplayer(thePipeOpti, listOfObjectives, theOptimizer, additionalWid-
                             gets=None)
    Class used to display optimization process in real time
    signal_optimization_over
    set_actionsOnClick(self, theList)
        Set actions to perform on click, list of on_graph_click_interface
    generate_optimizationGraphs(self, refresh_time=0.1)
        Generates the optimization graphs. :return: Graphs, LinkDataGraph,
        :class: ~optimeed.visulaize.gui.widgets.widget_graphs_visual.widget_graphs_visual
    create_main_window(self)
        From the widgets and the actions on click, spawn a window and put a gui around widgetsGraphsVisual.
    __change_appearance_violate_constraints(self)
    __auto_refresh(self, refresh_time)
    __set_graphs_disposition(self)
        Set nicely the graphs disposition
    launch_optimization(self)
        Perform the optimization and spawn the convergence graphs afterwards.
    __callback_optimization(self, myWindow)

class Worker
    Bases: PyQt5.QtCore.QObject
    signal_show_UI
    display_graphs(self, theGraphs)
```

fastPlot

Module Contents

```
class PlotHolders

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

```

    new_plot (self)
    set_title (self, theTitle, **kwargs)
    reset (self)
    axis_equal (self)
class WindowHolders

    set_currFigure (self, currFigure)
    add_plot (self, *args, **kwargs)
    set_title (self, *args, **kwargs)
    new_figure (self)
    new_plot (self)
    show (self)
    get_curr_plotHolder (self)
    get_wgGraphs (self, fig=None)
    get_all_figures (self)
    axis_equal (self)
myWindows
plot (x, y, hold=False, **kwargs)
    Plot new trace
show ()
    Show (start qt mainloop) graphs. Blocking
figure (numb)
    Set current figure
new_plot ()
    Add new plot
set_title (theTitle, **kwargs)
    Set title of the plot
axis_equal ()
get_all_figures ()
    Get all existing figures
get_wgGraphs (fig=None)
    Advanced option. :return: widget_graphs_visual

```

Package Contents

```

class gui_mainWindow (QtWidgetList, isLight=True, actionOnWindowClosed=None, neverCloseWin-
                        dow=False, title_window='Awesome Visualisation Tool', size=None)
    Bases: PyQt5.QtWidgets.QMainWindow
    Main class that spawns a Qt window. Use run () to display it.
    set_actionOnClose (self, actionOnWindowClosed)

```

```
closeEvent (self, event)  
run (self, hold=False)  
    Display the window  
keyPressEvent (self, event)  
  
app  
  
start_qt_mainloop ()  
    Starts qt mainloop, which is necessary for qt to handle events  
  
stop_qt_mainloop ()  
    Stops qt mainloop and resumes to program  
  
class gui_collection_exporter  
    Bases: PyQt5.QtWidgets.QMainWindow  
  
    Simple gui that allows to export data  
  
    signal_has_exported  
  
    signal_has_reset  
  
    exportCollection (self)  
        Export the collection  
  
    reset (self)  
  
    add_data_to_collection (self, data)  
        Add data to the collection to export  
  
        Parameters data – Whichever type you like  
  
    set_info (self, info)  
  
    set_collection (self, theCollection)  
  
class DataAnimationVisuals (id=0, window_title='Animation')  
    Bases: PyQt5.QtWidgets.QMainWindow  
  
    Spawns a gui that includes button to create animations nicely when paired with widget_graphs_visual  
  
    SLIDER_MAXIMUM_VALUE = 500  
  
    SLIDER_MINIMUM_VALUE = 1  
  
    add_trace (self, trace_id, element_list, theTrace)  
        Add a trace to the animation.  
  
        Parameters  
  
        • trace_id – id of the trace  
  
        • element_list – List of elements to save: [[OpenGL_item1, text_item1],  
            [OpenGL_item2, text_item2], ... [OpenGL_itemN, text_itemN]]  
  
        • theTrace – TraceVisual  
  
        Returns  
  
    add_elementToTrace (self, trace_id, indexPoint)  
  
    delete_point (self, trace_id, thePoint)  
  
    reset_all (self)  
  
    delete_all (self)
```

```

pause_play (self)
show_all (self)
next_frame (self)
slider_handler (self)
frame_selector (self)
set_refreshTime (self)
is_empty (self)
run (self)
closeEvent (self, _)
contains_trace (self, trace_id)
export_picture (self)
class widget_graphs_visual (theGraphs, **kwargs)
    Bases: PyQt5.QtWidgets.QWidget

    Widget element to draw a graph. The traces and graphs to draw are defined in Graphs taken as argument. This
    widget is linked to the excellent third-party library pyqtgraph, under MIT license

    signal_must_update
    signal_graph_changed
set_graph_disposition (self, indexGraph, row=1, col=1, rowspan=1, colspan=1)
    Change the graphs disposition.

        Parameters

- indexGraph – index of the graph to change
- row – row where to place the graph
- col – column where to place the graph
- rowspan – number of rows across which the graph spans
- colspan – number of columns across which the graph spans

Returns

__create_graph (self, idGraph)
__check_graphs (self)
on_click (self, plotDataItem, clicked_points)
update_graphs (self, singleUpdate=True)
    This method is used to update the graph. This is fast but NOT safe (especially when working with threads).
    To limit the risks, please use self.signal_must_update.emit() instead.

        Parameters singleUpdate – if set to False, the graph will periodically refres each
        self.refreshTime

fast_update (self)
    Use this method to update the graph in a fast way. NOT THREAD SAFE.

exportGraphs (self)
    Export the graphs

link_axes (self)

```

get_graph (*self*, *idGraph*)

Get corresponding GraphVisual of the graph *idGraph*

keyPressEvent (*self*, *event*)

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

delete_graph (*self*, *idGraph*)

Delete the graph *idGraph*

delete (*self*)

get_all_graphsVisual (*self*)

Return a dictionary {*idGraph*: GraphVisual}.

get_layout_buttons (*self*)

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

set_actionOnClick (*self*, *theActionOnClick*)

Action to perform when the graph is clicked

Parameters **theActionOnClick** – *on_graph_click_interface*

Returns

set_title (*self*, *idGraph*, *titleName*, ***kwargs*)

Set title of the graph

Parameters

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

set_article_template (*self*, *graph_size_x*=8.8, *graph_size_y*=4.4, *legendPosition*='NW')

Method to set the graphs to article quality graph.

Parameters

- **graph_size_x** – width of the graph in cm
- **graph_size_y** – height of the graph in cm
- **legendPosition** – position of the legend (NE, SE, SW, NW)

Returns

class widget_line_drawer (*minWinHeight*=300, *minWinWidth*=300, *is_light*=True)

Bases: PyQt5.QtWidgets.QWidget

Widget allowing to display several lines easily

signal_must_update

on_update_signal (*self*, *listOfLines*)

delete_lines (*self*, *key_id*)

Dele the lines :param *key_id*: id to delete :return:

set_lines (*self*, *listOfLines*, *key_id*=0, *pen*=None)

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

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

get_extrema_lines (*self*)

```

class widget_menuButton (theParentButton)
    Bases: PyQt5.QtWidgets.QMenu

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

    showEvent (self, QShowEvent)

class widget_opengl (parent=None)
    Bases: PyQt5.QtWidgets.QOpenGLWidget

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

    sizeHint (self)

    minimumSizeHint (self)

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

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

    initializeGL (self)

    paintGL (self)

    resizeGL (self, w, h)

    mousePressEvent (self, event)

    mouseMoveEvent (self, event)

    keyPressEvent (self, event)

    wheelEvent (self, QWheelEvent)

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

    Widget able to display a text

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

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

    refreshTraceList (self)
        Refresh all the traces

class DeviceDrawerInterface

    keyboard_push_action (self, theKey)

    get_colour_scalebar (self)

    get_colour_background (self)

    get_opengl_options (self)

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

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

```

```
    apply (self)
    reset (self)
    graph_clicked (self, theGraphVisual, index_graph, index_trace, indices_points)
    get_name (self)
class on_graph_click_export (theDataLink)
    Bases: optimeed.visualize.gui.widgets.widget_graphs_visual.
on_graph_click_interface
    On click: export the selected points
    graph_clicked (self, theGraphVisual, index_graph, index_trace, indices_points)
    reset_graph (self)
    get_name (self)
class on_click_extract_pareto (theDataLink, max_x=False, max_y=False)
    Bases: optimeed.visualize.gui.widgets.widget_graphs_visual.
on_graph_click_interface
    On click: extract the pareto from the cloud of points
    graph_clicked (self, the_graph_visual, index_graph, index_trace, _)
    get_name (self)
class on_graph_click_showInfo (theLinkDataGraph, visuals=None)
    Bases: optimeed.visualize.gui.widgets.widget_graphs_visual.
on_graph_click_interface
    On click: show informations about the points (loop through attributes)
class DataInformationVisuals

    delete_visual (self, theVisual)
    add_visual (self, theVisual, theTrace, indexPoint)
    get_new_index (self)
    curr_index (self)
    graph_clicked (self, theGraphVisual, index_graph, index_trace, indices_points)
        Action to perform when a point in the graph has been clicked: Creates new window displaying the device
        and its informations
    get_name (self)
class Repr_opengl (DeviceDrawer)

    get_widget (self, theNewDevice)
class Repr_lines (attribute_lines)

    get_widget (self, theNewDevice)
class on_graph_click_remove_trace (theDataLink)
    Bases: optimeed.visualize.gui.widgets.widget_graphs_visual.
on_graph_click_interface
```



```

graph_clicked (self, theGraphVisual, index_graph, index_trace, _)
get_name (self)
class on_click_copy_something (theDataLink, functionStrFromDevice)
    Bases: optimeed.visualize.gui.widgets.widget_graphs_visual.
            on_graph_click_interface
    On Click: copy something
    graph_clicked (self, the_graph_visual, index_graph, index_trace, indices_points)
    get_name (self)
class on_click_change_symbol (theLinkDataGraph)
    Bases: optimeed.visualize.gui.widgets.widget_graphs_visual.
            on_graph_click_interface
    On Click: Change the symbol of the point that is clicked
    graph_clicked (self, theGraphVisual, index_graph, index_trace, indices_points)
    get_name (self)
class on_graph_click_interface
    Interface class for the action to perform when a point is clicked
class DataAnimationOpenGL (theOpenGLWidget, theId=0, window_title='Animation')
    Bases: optimeed.visualize.gui.gui_data_animation.DataAnimationVisuals
    Implements DataAnimationVisuals to show opengl drawing
    update_widget_w_animation (self, key, index, the_data_animation)
    export_widget (self, painter)
    delete_key_widgets (self, key)
class DataAnimationOpenGLwText (*args, is_light=True, **kwargs)
    Bases: optimeed.visualize.gui.widgets.graphsVisualWidget.
            examplesActionOnClick.on_click_anim.DataAnimationOpenGL
    Implements DataAnimationVisuals to show opengl drawing and text
    update_widget_w_animation (self, key, index, the_data_animation)
    get_interesting_elements (self, devices_list)
class DataAnimationLines (is_light=True, theId=0, window_title='Animation')
    Bases: optimeed.visualize.gui.gui_data_animation.DataAnimationVisuals
    Implements DataAnimationVisuals to show drawing made out of lines (widget_line_drawer)
    export_widget (self, painter)
    delete_key_widgets (self, key)
    update_widget_w_animation (self, key, index, the_data_animation)
    get_interesting_elements (self, devices_list)
class DataAnimationVisualswText (is_light=True, theId=0, window_title='Animation')
    Bases: optimeed.visualize.gui.widgets.graphsVisualWidget.
            examplesActionOnClick.on_click_anim.DataAnimationLines
    Same as DataAnimationLines but also with text
    update_widget_w_animation (self, key, index, the_data_animation)

```

```
class on_graph_click_showAnim(theLinkDataGraph, theAnimation)
    Bases: optimeed.visualize.gui.widgets.widget_graphs_visual.
            on_graph_click_interface

    On click: add or remove an element to animate

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

    get_name(self)

class LinkDataGraph

    class _collection_linker

        add_link(self, idSlave, idMaster)

        get_collection_master(self, idToGet)

        is_slave(self, idToCheck)

        set_same_master(self, idExistingSlave, idOtherSlave)
            Parameters
            • idExistingSlave – id collection of the existing slave
            • idOtherSlave – id collection of the new slave that has to be linked to an existing
              master

        add_collection(self, theCollection, kwargs=None)

        add_graph(self, howToPlotGraph)

        createGraphs(self)

        get_howToPlotGraph(self, idGraph)

        get_collectionInfo(self, idCollectionInfo)

        create_trace(self, collectionInfo, howToPlotGraph, idGraph)

        get_all_id_graphs(self)

        get_all_traces_id_graph(self, idGraph)

        update_graphs(self)

        is_slave(self, idGraph, idTrace)

        get_idCollection_from_graph(self, idGraph, idTrace, getMaster=True)
            From indices in the graph, get index of corresponding collection

        get_collection_from_graph(self, idGraph, idTrace, getMaster=True)
            From indices in the graph, get corresponding collection

        get_dataObject_from_graph(self, idGraph, idTrace, idPoint)

        get_dataObjects_from_graph(self, idGraph, idTrace, idPoint_list)

        remove_element_from_graph(self, idGraph, idTrace, idPoint, deleteFromMaster=False)
            Remove element from the graph, or the master collection

        remove_elements_from_trace(self, idGraph, idTrace, idPoints, deleteFromMaster=False)
            Performances      optimisation      when      compared      to      LinkDataGraph.
            remove_element_from_graph()

        link_collection_to_graph_collection(self, id_collection_graph, id_collection_master)
            Link data :param id_collection_graph: :param id_collection_master: :return:
```

```

remove_trace (self, idGraph, idTrace)

get_graph_and_trace_from_collection (self, idCollection)
    Reverse search: from a collection, get the associated graph

get_mappingData_graph (self, idGraph)

get_mappingData_trace (self, idGraph, idTrace)

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

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

    __str__ (self)

class gui_mainWindow (QtWidgetList, isLight=True, actionOnWindowClosed=None, neverCloseWin-
                        dow=False, title_window='Awesome Visualisation Tool', size=None)
    Bases: PyQt5.QtWidgets.QMainWindow

    Main class that spawns a Qt window. Use run() to display it.

    set_actionOnClose (self, actionOnWindowClosed)

    closeEvent (self, event)

    run (self, hold=False)
        Display the window

    keyPressEvent (self, event)

class widget_graphs_visual (theGraphs, **kwargs)
    Bases: PyQt5.QtWidgets.QWidget

    Widget element to draw a graph. The traces and graphs to draw are defined in Graphs taken as argument. This
    widget is linked to the excellent third-party library pyqtgraph, under MIT license

    signal_must_update

    signal_graph_changed

    set_graph_disposition (self, indexGraph, row=1, col=1, rowspan=1, colspan=1)
        Change the graphs disposition.

        Parameters

        • indexGraph – index of the graph to change

        • row – row where to place the graph

        • col – column where to place the graph

        • rowspan – number of rows across which the graph spans

        • colspan – number of columns across which the graph spans

        Returns

    __create_graph (self, idGraph)

    __check_graphs (self)

    on_click (self, plotDataItem, clicked_points)

    update_graphs (self, singleUpdate=True)
        This method is used to update the graph. This is fast but NOT safe (especially when working with threads).
        To limit the risks, please use self.signal_must_update.emit() instead.

```

Parameters `singleUpdate` – if set to False, the graph will periodically refresh each `self.refreshTime`

fast_update (*self*)

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

exportGraphs (*self*)

Export the graphs

link_axes (*self*)

get_graph (*self*, *idGraph*)

Get corresponding `GraphVisual` of the graph `idGraph`

keyPressEvent (*self*, *event*)

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

delete_graph (*self*, *idGraph*)

Delete the graph `idGraph`

delete (*self*)

get_all_graphsVisual (*self*)

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

get_layout_buttons (*self*)

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

set_actionOnClick (*self*, *theActionOnClick*)

Action to perform when the graph is clicked

Parameters `theActionOnClick` – *on_graph_click_interface*

Returns

set_title (*self*, *idGraph*, *titleName*, ***kwargs*)

Set title of the graph

Parameters

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

set_article_template (*self*, *graph_size_x*=8.8, *graph_size_y*=4.4, *legendPosition*='NW')

Method to set the graphs to article quality graph.

Parameters

- **graph_size_x** – width of the graph in cm
- **graph_size_y** – height of the graph in cm
- **legendPosition** – position of the legend (NE, SE, SW, NW)

Returns

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

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

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

class `DataInformationVisuals`

delete_visual (*self*, *theVisual*)

```

    add_visual (self, theVisual, theTrace, indexPoint)

    get_new_index (self)

    curr_index (self)

graph_clicked (self, theGraphVisual, index_graph, index_trace, indices_points)
    Action to perform when a point in the graph has been clicked: Creates new window displaying the device
    and its informations

get_name (self)

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

    refreshTraceList (self)
        Refresh all the traces

class OptimizationDisplayer (thePipeOpti, listOfObjectives, theOptimizer, additionalWid-
                                gets=None)
    Class used to display optimization process in real time

    signal_optimization_over

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

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

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

    __change_appearance_violate_constraints (self)

    __auto_refresh (self, refresh_time)

    __set_graphs_disposition (self)
        Set nicely the graphs disposition

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

    __callback_optimization (self, myWindow)

class Worker
    Bases: PyQt5.QtCore.QObject

    signal_show_UI

    display_graphs (self, theGraphs)

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_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 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 = 1.1.0
```


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 ansi2html](#), [21](#)
[optimeed core ansi2html converter](#), [21](#)
[optimeed core ansi2html style](#), [23](#)
[optimeed core ansi2html util](#), [23](#)
[optimeed core collection](#), [24](#)
[optimeed core color_palette](#), [26](#)
[optimeed core commonImport](#), [26](#)
[optimeed core graphs](#), [26](#)
[optimeed core interfaceDevice](#), [30](#)
[optimeed core linkDataGraph](#), [30](#)
[optimeed core myjson](#), [31](#)
[optimeed core options](#), [32](#)
[optimeed core tools](#), [33](#)
[optimeed optimize](#), [45](#)
[optimeed optimize characterization](#), [46](#)
[optimeed optimize characterization characterization](#), [46](#)
[optimeed optimize characterization interfaceCharacterization](#), [46](#)
[optimeed optimize mathsToPhysics](#), [46](#)
[optimeed optimize mathsToPhysics interfaceMathsToPhysics](#), [46](#)
[optimeed optimize mathsToPhysics mathsToPhysics](#), [47](#)
[optimeed optimize objAndCons](#), [47](#)
[optimeed optimize objAndCons fastObjCons](#), [47](#)
[optimeed optimize objAndCons interfaceObjCons](#), [47](#)
[optimeed optimize optiAlgorithms](#), [48](#)
[optimeed optimize optiAlgorithms algorithmInterface](#), [51](#)
[optimeed optimize optiAlgorithms convergence](#), [48](#)
[optimeed optimize optiAlgorithms convergence evolution](#), [48](#)
[optimeed optimize optiAlgorithms convergence hyper](#), [49](#)
[optimeed optimize optiAlgorithms convergence inter](#), [50](#)
[optimeed optimize optiAlgorithms multiObjective_GA](#), [51](#)
[optimeed optimize optiAlgorithms NLOpt_Algorithm](#), [51](#)
[optimeed optimize optimizer](#), [54](#)
[optimeed optimize optiVariable](#), [53](#)
[optimeed visualize](#), [59](#)
[optimeed visualize displayOptimization](#), [94](#)
[optimeed visualize fastPlot](#), [94](#)
[optimeed visualize gui](#), [59](#)
[optimeed visualize gui gui_collection_exporter](#), [84](#)
[optimeed visualize gui gui_data_animation](#), [84](#)
[optimeed visualize gui gui_data_selector](#), [85](#)
[optimeed visualize gui gui_mainWindow](#), [86](#)
[optimeed visualize gui widgets](#), [59](#)
[optimeed visualize gui widgets graphsVisualWidget](#), [59](#)
[optimeed visualize gui widgets graphsVisualWidget](#), [59](#)
[optimeed visualize gui widgets graphsVisualWidget](#), [60](#)
[optimeed visualize gui widgets graphsVisualWidget](#), [60](#)
[optimeed visualize gui widgets graphsVisualWidget](#), [60](#)
[optimeed visualize gui widgets graphsVisualWidget](#), [60](#)

61
optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick.on_click_extract_p
61
optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick.on_click_remove_tr
61
optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick.on_click_showinfo,
62
optimeed.visualize.gui.widgets.graphsVisualWidget.graphVisual,
66
optimeed.visualize.gui.widgets.graphsVisualWidget.pyqtgraphRedefine,
67
optimeed.visualize.gui.widgets.graphsVisualWidget.smallGui,
69
optimeed.visualize.gui.widgets.graphsVisualWidget.traceVisual,
69
optimeed.visualize.gui.widgets.openGLWidget,
71
optimeed.visualize.gui.widgets.openGLWidget.ContextHandler,
71
optimeed.visualize.gui.widgets.openGLWidget.DeviceDrawerInterface,
72
optimeed.visualize.gui.widgets.openGLWidget.Materials_visual,
72
optimeed.visualize.gui.widgets.openGLWidget.OpenGlFunctions_Library,
73
optimeed.visualize.gui.widgets.openGLWidget.quaternions,
74
optimeed.visualize.gui.widgets.openGLWidget.TriangulatePolygon,
74
optimeed.visualize.gui.widgets.widget_graphs_visual,
74
optimeed.visualize.gui.widgets.widget_line_drawer,
76
optimeed.visualize.gui.widgets.widget_menuButton,
76
optimeed.visualize.gui.widgets.widget_openGL,
77
optimeed.visualize.gui.widgets.widget_text,
77

Symbols

- `__CONVERGENCE` (*OptiHistoric* attribute), 55
 - `__DATA_STR` (*ListDataStruct* attribute), 25, 36
 - `__DEVICE` (*OptiHistoric* attribute), 54
 - `__INFO_STR` (*ListDataStruct* attribute), 25, 36
 - `__LOGOPTI` (*OptiHistoric* attribute), 54
 - `__RESULTS` (*OptiHistoric* attribute), 54
 - `__State` (class in *optimeed.core.ansi2html.converter*), 22
 - `__amb3__` (*MaterialRenderingProperties* attribute), 73
 - `__author__` (in module *optimeed.optimize.optiAlgorithms.convergence.hypervolume*), 49
 - `__auto_refresh()` (*OptimizationDisplayer* method), 94, 105
 - `__callback_optimization()` (*OptimizationDisplayer* method), 94, 105
 - `__change_appearance_violate_constraints()` (*OptimizationDisplayer* method), 94, 105
 - `__check_graphs()` (*widget_graphs_visual* method), 75, 78, 88, 97, 103, 106
 - `__create_graph()` (*widget_graphs_visual* method), 75, 78, 88, 97, 103, 106
 - `__dif3__` (*MaterialRenderingProperties* attribute), 72
 - `__draw_axis__()` (*ContextHandler* method), 72
 - `__len__()` (*MultiList* method), 50
 - `__lightingInit__()` (*ContextHandler* method), 72
 - `__reset__()` (*ContextHandler* method), 72
 - `__set_graphs_disposition()` (*OptimizationDisplayer* method), 94, 105
 - `__shin__` (*MaterialRenderingProperties* attribute), 73
 - `__spec3__` (*MaterialRenderingProperties* attribute), 72
 - `__str__()` (*Attribute_selector* method), 86
 - `__str__()` (*AutosaveStruct* method), 24, 36
 - `__str__()` (*Binary_OptimizationVariable* method), 54, 57
 - `__str__()` (*Container_attribute_selector* method), 86
 - `__str__()` (*Data* method), 28, 41, 109
 - `__str__()` (*DataStruct_Interface* method), 24, 36
 - `__str__()` (*HowToPlotGraph* method), 30, 43, 103
 - `__str__()` (*Integer_OptimizationVariable* method), 54, 58
 - `__str__()` (*InterfaceCharacterization* method), 46, 56
 - `__str__()` (*InterfaceObjCons* method), 48, 57
 - `__str__()` (*MathsToPhysics* method), 47, 56
 - `__str__()` (*MultiList* method), 49
 - `__str__()` (*MultiList.Node* method), 49
 - `__str__()` (*MultiObjective_GA* method), 52, 53, 57
 - `__str__()` (*NLOpt_Algorithm* method), 51
 - `__str__()` (*OptimizationVariable* method), 53
 - `__str__()` (*Options* method), 32, 45
 - `__str__()` (*Real_OptimizationVariable* method), 53, 57
 - `__str__()` (*Rule* method), 23
 - `__apply_regex()` (*Ansi2HTMLConverter* method), 23, 24
 - `__collapse_cursor()` (*Ansi2HTMLConverter* method), 23, 24
 - `__find_class()` (in module *optimeed.core.myjson*), 32
 - `__get_object_class()` (in module *optimeed.core.myjson*), 31
 - `__get_object_module()` (in module *optimeed.core.myjson*), 31
 - `__html_template` (in module *optimeed.core.ansi2html.converter*), 22
 - `__instantiates_annotated_object()` (in module *optimeed.core.myjson*), 32
 - `__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*), 31
 - `__remove_index_from_show()` (*DataAnimationTrace* method), 84
- ## A
- `Action_on_selector_update` (class in *optimeed.visualize.gui.gui_data_selector*), 85

activateMaterialProperties() (*MaterialRenderingProperties* method), 73
 add_attribute_selector() (*Container_attribute_selector* method), 86
 add_child() (*Attribute_selector* method), 85
 add_child() (*Container_attribute_selector* method), 86
 add_collection() (*LinkDataGraph* method), 30, 44, 102
 add_data() (*GraphVisual* method), 67
 add_data() (*ListDataStruct* method), 25, 36
 add_data_to_collection() (*gui_collection_exporter* method), 84, 87, 96
 add_element() (*DataAnimationTrace* method), 84
 add_elementToTrace() (*DataAnimationVisuals* method), 64, 82, 85, 88, 92, 96
 add_feature() (*GraphVisual* method), 66
 add_graph() (*Graphs* method), 29, 43, 110
 add_graph() (*LinkDataGraph* method), 30, 44, 102
 add_index_to_show() (*DataAnimationTrace* method), 84
 add_link() (*LinkDataGraph_collection_linker* method), 30, 44, 102
 add_modified_paintElem() (*TraceVisual._ModifiedPaintElem* method), 69
 add_option() (*Option_class* method), 20, 33, 45
 add_option() (*Options* method), 32, 45
 add_plot() (*PlotHolders* method), 94, 110
 add_plot() (*WindowHolders* method), 95, 110
 add_point() (*ConvergenceManager* method), 51
 add_point() (*Data* method), 28, 41, 109
 add_point() (*OptiHistoric* method), 55
 add_suffix_to_path() (*in module optimeed.core*), 38
 add_suffix_to_path() (*in module optimeed.core.tools*), 33
 add_trace() (*DataAnimationVisuals* method), 64, 81, 85, 87, 92, 96
 add_trace() (*Graph* method), 28, 42
 add_trace() (*Graphs* method), 29, 42, 109
 add_trace() (*GraphVisual* method), 67
 add_trace_firstGraph() (*Graphs* method), 28, 42, 109
 add_update_method() (*Graphs* method), 29, 43, 110
 add_visual() (*on_graph_click_showInfo.DataInformationVisuals* method), 62, 63, 80, 91, 100, 104
 addItem() (*myGraphicsLayout* method), 68
 addItem() (*myLegend* method), 68
 adjust() (*_State* method), 22
 ALGORITHM (*NLOpt_Algorithm* attribute), 51
 AlgorithmInterface (class *in optimeed.optimize.optiAlgorithms.algorithmInterface*), 51
 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

meed.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*), 96
 app (in module *optimeed.visualize.gui*), 87
 app (in module *optimeed.visualize.gui.gui_data_selector*), 85
 app (in module *optimeed.visualize.gui.gui_mainWindow*), 86
 append() (*MultiList* method), 50
 apply() (*on_graph_click_delete* method), 61, 62, 80, 91, 99
 apply_filters() (*GuiDataSelector* method), 86
 apply_palette() (*GraphVisual* method), 66
 apply_regex() (*Ansi2HTMLConverter* method), 23, 24
 apply_width_sample() (*myLegend* method), 68
 applyEquation() (in module *optimeed.core*), 35, 38
 applyEquation() (in module *optimeed.core.tools*), 33
 arithmeticEval() (in module *optimeed.core*), 38
 arithmeticEval() (in module *optimeed.core.tools*), 33
 assign() (*InterfaceDevice* method), 30, 43
 Attribute_selector (class in *optimeed.visualize.gui.gui_data_selector*), 85
 attrs() (*Ansi2HTMLConverter* method), 23, 24
 AutosaveStruct (class in *optimeed.core*), 36
 AutosaveStruct (class in *optimeed.core.collection*), 24
 axis_equal() (*GraphVisual* method), 67
 axis_equal() (in module *optimeed.visualize*), 111
 axis_equal() (in module *optimeed.visualize.fastPlot*), 95
 axis_equal() (*PlotHolders* method), 95, 110
 axis_equal() (*WindowHolders* method), 95, 111
 axisangle_to_q() (in module *optimeed.visualize.gui.widgets.openGLWidget.quaternions*), 74

B

Binary_OptimizationVariable (class in *optimeed.optimize*), 57
 Binary_OptimizationVariable (class in *optimeed.optimize.optiVariable*), 54
 blackOnly() (in module *optimeed.core.color_palette*), 26
 BLUE (*text_format* attribute), 20, 33, 37, 45

Blue_material (in module *optimeed.visualize.gui.widgets.openGLWidget.Materials_visual*), 73
 BOLD (*text_format* attribute), 20, 33, 37, 45
 Brass_material (in module *optimeed.visualize.gui.widgets.openGLWidget.Materials_visual*), 73
 Bronze_material (in module *optimeed.visualize.gui.widgets.openGLWidget.Materials_visual*), 73

C

callback_on_evaluation() (*Optimizer* method), 56, 58
 cart2pol() (in module *optimeed.core*), 38
 cart2pol() (in module *optimeed.core.tools*), 33
 Characterization (class in *optimeed.optimize*), 56
 Characterization (class in *optimeed.optimize.characterization*), 46
 Characterization (class in *optimeed.optimize.characterization.characterization*), 46
 check_and_add_if_float() (in module *optimeed.visualize.gui.gui_data_selector*), 86
 Chrome_material (in module *optimeed.visualize.gui.widgets.openGLWidget.Materials_visual*), 73
 CLASS_TAG (in module *optimeed.core.myjson*), 31
 CLIC_LEFT (in module *optimeed.visualize.gui.widgets.openGLWidget.ContextHandler*), 71
 CLIC_RIGHT (in module *optimeed.visualize.gui.widgets.openGLWidget.ContextHandler*), 71
 close() (*MyMultiprocessEvaluator* method), 52
 closeEvent() (*DataAnimationVisuals* method), 64, 82, 85, 88, 93, 97
 closeEvent() (*gui_mainWindow* method), 86, 87, 95, 103, 107
 CollectionInfo (class in *optimeed.core*), 43
 CollectionInfo (class in *optimeed.core.linkDataGraph*), 30
 color() (in module *optimeed.core.ansi2html.style*), 23
 color_component() (in module *optimeed.core.ansi2html.style*), 23
 compute() (*Characterization* method), 46, 56
 compute() (*FastObjCons* method), 47, 48, 56
 compute() (*HyperVolume* method), 49
 compute() (*MultiObjective_GA* method), 52, 53, 57
 compute() (*NLOpt_Algorithm* method), 51
 constraints (*OptiHistoric._pointData* attribute), 54
 constraints_per_step (*EvolutionaryConvergence* attribute), 48, 50

Container_attribute_selector (class in *optimeed.visualize.gui.gui_data_selector*), 86
 contains_trace() (*DataAnimationVisuals* method), 64, 82, 85, 88, 93, 97
 ContextHandler (class in *optimeed.visualize.gui.widgets.openGLWidget.ContextHandler*), 72
 conv (*MyConvergence* attribute), 51
 ConvergenceManager (class in *optimeed.optimize.optiAlgorithms.NLOpt_Algorithm*), 51
 convert() (*Ansi2HTMLConverter* method), 23, 24
 Copper_material (in module *optimeed.visualize.gui.widgets.openGLWidget.Materials_visual*), 73
 copy() (*Options* method), 32, 45
 create_main_window() (*OptimizationDisplayer* method), 94, 105
 create_trace() (*LinkDataGraph* method), 31, 44, 102
 create_unique_dirname() (in module *optimeed.core*), 38
 create_unique_dirname() (in module *optimeed.core.tools*), 33
 createGraphs() (*LinkDataGraph* method), 31, 44, 102
 curr_index() (*on_graph_click_showInfo.DataInformationVisuals* method), 62, 63, 80, 91, 100, 105
 CursorMoveUp (class in *optimeed.core.ansi2html.converter*), 22
 CYAN (*text_format* attribute), 20, 33, 37, 45
D
 dark2() (in module *optimeed.core.color_palette*), 26
 DARKCYAN (*text_format* attribute), 20, 33, 37, 45
 Data (class in *optimeed.core*), 39
 Data (class in *optimeed.core.graphs*), 26
 Data (class in *optimeed.visualize*), 107
 DataAnimationLines (class in *optimeed.visualize*), 101
 DataAnimationLines (class in *optimeed.visualize.gui*), 93
 DataAnimationLines (class in *optimeed.visualize.gui.widgets*), 83
 DataAnimationLines (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick*), 65
 DataAnimationLines (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick.on_click_anim*), 59
 DataAnimationOpenGL (class in *optimeed.visualize*), 101
 DataAnimationOpenGL (class in *optimeed.visualize.gui*), 93
 DataAnimationOpenGL (class in *optimeed.visualize.gui.widgets*), 82
 DataAnimationOpenGL (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick*), 65
 DataAnimationOpenGL (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick.on_click_anim*), 59
 DataAnimationOpenGLwText (class in *optimeed.visualize*), 101
 DataAnimationOpenGLwText (class in *optimeed.visualize.gui*), 93
 DataAnimationOpenGLwText (class in *optimeed.visualize.gui.widgets*), 83
 DataAnimationOpenGLwText (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick*), 65
 DataAnimationOpenGLwText (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick.on_click_anim*), 59
 DataAnimationTrace (class in *optimeed.visualize.gui.gui_data_animation*), 84
 DataAnimationTrace.element_animation (class in *optimeed.visualize.gui.gui_data_animation*), 84
 DataAnimationVisuals (class in *optimeed.visualize*), 96
 DataAnimationVisuals (class in *optimeed.visualize.gui*), 87, 92
 DataAnimationVisuals (class in *optimeed.visualize.gui.gui_data_animation*), 85
 DataAnimationVisuals (class in *optimeed.visualize.gui.widgets*), 81
 DataAnimationVisuals (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick*), 64
 DataAnimationVisualswText (class in *optimeed.visualize*), 101
 DataAnimationVisualswText (class in *optimeed.visualize.gui*), 93
 DataAnimationVisualswText (class in *optimeed.visualize.gui.widgets*), 83
 DataAnimationVisualswText (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick*), 65
 DataAnimationVisualswText (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick.on_click_anim*), 59
 DataStruct_Interface (class in *optimeed.core*), 36
 DataStruct_Interface (class in *optimeed*), 36

meed.core.collection), 24
 decode_str_json() (in module optimeed.core), 35
 decode_str_json() (in module optimeed.core.myjson), 32
 default (in module optimeed.optimize.optimizer), 54
 default_palette() (in module optimeed.core.color_palette), 26
 delete() (GraphVisual method), 67
 delete() (widget_graphs_visual method), 75, 78, 89, 98, 104, 106
 delete_all() (DataAnimationTrace method), 84
 delete_all() (DataAnimationVisuals method), 64, 82, 85, 88, 93, 96
 delete_graph() (widget_graphs_visual method), 75, 78, 89, 98, 104, 106
 delete_gui (class in optimeed.visualize.gui.widgets.graphsVisualWidget.examples.ActionsOnClick.onClick_delete), module optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctions_Lib, 60
 delete_indices_from_list() (in module optimeed.core), 39
 delete_indices_from_list() (in module optimeed.core.tools), 35
 delete_key_widgets() (DataAnimationLines method), 59, 65, 83, 93, 101
 delete_key_widgets() (DataAnimationOpenGL method), 59, 65, 83, 93, 101
 delete_lines() (widget_line_drawer method), 65, 76, 79, 82, 90, 98
 delete_point() (Data method), 28, 41, 109
 delete_point() (DataAnimationVisuals method), 64, 82, 85, 88, 92, 96
 delete_points_at_indices() (ListDataStruct method), 25, 37
 delete_trace() (GraphVisual method), 67
 delete_visual() (on_graph_click_showInfo.DataAnimationVisuals method), 62, 63, 80, 91, 100, 104
 derivate() (in module optimeed.core), 39
 derivate() (in module optimeed.core.tools), 35
 DeviceDrawerInterface (class in optimeed.visualize), 99
 DeviceDrawerInterface (class in optimeed.visualize.gui), 90
 DeviceDrawerInterface (class in optimeed.visualize.gui.widgets), 83
 DeviceDrawerInterface (class in optimeed.visualize.gui.widgets.openGLWidget.DeviceDrawerInterface), 72
 display_graphs() (Worker method), 94, 105
 DISPLAY_INFO (Optimizer attribute), 55, 58
 dist() (in module optimeed.core), 38
 dist() (in module optimeed.core.tools), 34
 DIVISION_OUTER (MultiObjective_GA attribute), 52, 53, 57
 do_MathsToPhys() (Binary_OptimizationVariable method), 54, 57
 do_MathsToPhys() (Integer_OptimizationVariable method), 54, 58
 do_MathsToPhys() (OptimizationVariable method), 53
 do_MathsToPhys() (Real_OptimizationVariable method), 53, 57
 draw_2Dring() (in module optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctions_Lib, 73
 draw_2Dring_diff_angle() (in module optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctions_Lib, 73
 draw_carved_disk() (in module optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctions_Lib, 74
 draw_cylinder() (in module optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctions_Lib, 73
 draw_disk() (in module optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctions_Lib, 74
 draw_extrudeZ() (in module optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctions_Lib, 73
 draw_lines() (in module optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctions_Lib, 73
 draw_part_cylinder() (in module optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctions_Lib, 73
 draw_part_cylinder_throat() (in module optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctions_Lib, 74
 draw_part_disk() (in module optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctions_Lib, 74
 draw_part_disk_diff_angles() (in module optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctions_Lib, 74
 draw_rectangle() (in module optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctions_Lib, 73
 draw_simple_rectangle() (in module optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctions_Lib, 73
 draw_spiral() (in module optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctions_Lib, 73
 draw_spiralFront() (in module opti-

meed.visualize.gui.widgets.openGLWidget.OpenGLFunctionsLibrary), (*widget_graphs_visual* method), 75, 73
 78, 89, 97, 104, 106
draw_spiralFull() (in module *optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctionsLibrary*), (*MyConvergence* method), 52
 73
draw_spiralSheet() (in module *optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctionsLibrary*), (*MyConvergence* method), 52
 73
draw_triList() (in module *optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctionsLibrary*), (*MyConvergence* method), 52
 73
draw_tubeSheet() (in module *optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctionsLibrary*), (*MyConvergence* method), 52
 73
drawWireTube() (in module *optimeed.visualize.gui.widgets.openGLWidget.OpenGLFunctionsLibrary*), (*MyConvergence* method), 52
 74

E

Emerald_material (in module *optimeed.visualize.gui.widgets.openGLWidget.MaterialsVisual*), (*MyConvergence* method), 52
 73
encode_str_json() (in module *optimeed.core*), 35
encode_str_json() (in module *optimeed.core.myjson*), 32
END (*text_format* attribute), 20, 33, 37, 45
evaluate() (*MyProblem* method), 52
evaluate() (*Parametric_analysis* method), 20, 21
evaluate_all() (*MyMapEvaluator* method), 52
evaluate_all() (*MyMultiprocessEvaluator* method), 52
evaluateObjectiveAndConstraints() (*Optimizer* method), 55, 58
EvolutionaryConvergence (class in *optimeed.optimize.optiAlgorithms.convergence*), 50
EvolutionaryConvergence (class in *optimeed.optimize.optiAlgorithms.convergence.evolutionaryConvergence*), 48
exclude_col() (*HowToPlotGraph* method), 30, 43, 103
EXCLUDED_TAGS (in module *optimeed.core.myjson*), 31
export_picture() (*DataAnimationVisuals* method), 64, 82, 85, 88, 93, 97
export_str() (*Data* method), 28, 41, 109
export_str() (*Graph* method), 28, 42
export_str() (*Graphs* method), 29, 43, 110
export_widget() (*DataAnimationLines* method), 59, 65, 83, 93, 101
export_widget() (*DataAnimationOpenGL* method), 59, 65, 82, 93, 101
export_xls() (*ListDataStruct* method), 25, 37
exportCollection() (*gui_collection_exporter* method), 84, 87, 96

F

fast_LUT_interpolation (class in *optimeed.core*), 39
fast_LUT_interpolation (class in *optimeed.core.tools*), 35
fast_update() (*GraphVisual* method), 67
fast_update() (*widget_graphs_visual* method), 75, 8, 9, 104, 106
FastObjCons (class in *optimeed.optimize*), 56
FastObjCons (class in *optimeed.optimize.objAndCons*), 48
FastObjCons (class in *optimeed.optimize.objAndCons.fastObjCons*), 47

G

figure() (in module *optimeed.visualize*), 111
figure_visual() (in module *optimeed.visualize.fastPlot*), 95
find_and_replace() (in module *optimeed.core*), 38
find_and_replace() (in module *optimeed.core.tools*), 33
formatInfo() (*Optimizer* method), 56, 58
frame_selector() (*DataAnimationVisuals* method), 64, 82, 85, 88, 93, 97
fromMathsToPhys() (*MathsToPhysics* method), 47, 56
fromPhysToMaths() (*MathsToPhysics* method), 47, 56

get_all_options() (*Option_class method*), 20, 32, 45
 get_all_traces() (*Graph method*), 28, 42
 get_all_traces() (*GraphVisual method*), 67
 get_all_traces_id_graph() (*LinkDataGraph method*), 31, 44, 102
 get_analyzed_attribute() (*Parametric_parameter method*), 19, 21
 get_attr_object() (in module *optimeed.visualize.gui.gui_data_selector*), 86
 get_attribute_name() (*OptimizationVariable method*), 53
 get_attribute_selectors() (*Container_attribute_selector method*), 86
 get_axis() (*GraphVisual method*), 66
 get_base_pen() (*DataAnimationTrace method*), 84
 get_base_pen() (*TraceVisual method*), 70
 get_base_symbol() (*TraceVisual method*), 70
 get_base_symbol_brush() (*TraceVisual method*), 70
 get_base_symbol_pen() (*TraceVisual method*), 70
 get_brushes() (*TraceVisual method*), 70
 get_children() (*Attribute_selector method*), 86
 get_children() (*Container_attribute_selector method*), 86
 get_collection() (*CollectionInfo method*), 30, 43
 get_collection_from_graph() (*LinkData-Graph method*), 31, 44, 102
 get_collection_master() (*LinkData-Graph_collection_linker method*), 30, 44, 102
 get_collectionInfo() (*LinkDataGraph method*), 31, 44, 102
 get_color() (*Data method*), 27, 40, 108
 get_color() (*TraceVisual method*), 70
 get_colour_background() (*DeviceDrawerInterface method*), 72, 83, 91, 99
 get_colour_scalebar() (*DeviceDrawerInterface method*), 72, 83, 91, 99
 get_convergence() (*MultiObjective_GA method*), 52, 53, 57
 get_convergence() (*NLOpt_Algorithm method*), 51
 get_convergence() (*OptiHistoric method*), 55
 get_curr_plotHolder() (*WindowHolders method*), 95, 111
 get_data() (*ListDataStruct method*), 25, 36
 get_data() (*TraceVisual method*), 70
 get_dataIndex_from_graphIndex() (*Data method*), 27, 40, 108
 get_dataIndices_from_graphIndices() (*Data method*), 27, 40, 108
 get_dataObject_from_graph() (*LinkData-Graph method*), 31, 44, 102
 get_dataObjects_from_graph() (*LinkData-Graph method*), 31, 44, 102
 get_datastruct() (*AutosaveStruct method*), 25, 36
 get_device() (*PipeOptimization method*), 54
 get_devices() (*OptiHistoric method*), 55
 get_element_animations() (*DataAnimation-Trace method*), 84
 get_extrema_lines() (*widget_line_drawer method*), 65, 76, 79, 82, 90, 98
 get_filename() (*AutosaveStruct method*), 24, 36
 get_first_graph() (*Graphs method*), 29, 42, 109
 get_graph() (*Graphs method*), 29, 42, 109
 get_graph() (*widget_graphs_visual method*), 75, 78, 89, 97, 104, 106
 get_graph_and_trace_from_collection() (*LinkDataGraph method*), 31, 44, 103
 get_graphIndex_from_dataIndex() (*Data method*), 27, 41, 108
 get_graphIndices_from_dataIndices() (*Data method*), 27, 41, 108
 get_graphs() (*EvolutionaryConvergence method*), 49, 51
 get_graphs() (*MyConvergence method*), 52
 get_historic() (*PipeOptimization method*), 54
 get_howToPlotGraph() (*LinkDataGraph method*), 31, 44, 102
 get_hypervolume_convergence() (*EvolutionaryConvergence method*), 49, 50
 get_id() (*CollectionInfo method*), 30, 44
 get_idCollection_from_graph() (*LinkData-Graph method*), 31, 44, 102
 get_indices_points_to_plot() (*Data method*), 28, 41, 109
 get_indices_to_show() (*DataAnimationTrace method*), 84
 get_info() (*DataStruct_Interface method*), 24, 36
 get_interesting_elements() (*DataAnimation-Lines method*), 59, 65, 83, 93, 101
 get_interesting_elements() (*DataAnimationOpenGLwText method*), 59, 65, 83, 93, 101
 get_invert_permutations() (*Data method*), 27, 40, 108
 get_kwargs() (*CollectionInfo method*), 30, 44
 get_label_pos() (*myAxis method*), 69
 get_last_pareto() (*EvolutionaryConvergence method*), 48, 50
 get_layout_buttons() (*widget_graphs_visual method*), 75, 78, 89, 98, 104, 106
 get_legend() (*Data method*), 28, 41, 108
 get_legend() (*GraphVisual method*), 66
 get_length() (*TraceVisual method*), 70
 get_length_data() (*Data method*), 26, 40, 107
 get_linestyle() (*Data method*), 28, 41, 109
 get_list_attributes() (*ListDataStruct method*),

- 25, 37
- `get_logopti()` (*OptiHistoric method*), 55
- `get_mappingData_graph()` (*LinkDataGraph method*), 31, 45, 103
- `get_mappingData_trace()` (*LinkDataGraph method*), 31, 45, 103
- `get_max_value()` (*Integer_OptimizationVariable method*), 54, 57
- `get_max_value()` (*Real_OptimizationVariable method*), 53, 57
- `get_min_max_attributes()` (*Attribute_selector method*), 86
- `get_min_value()` (*Integer_OptimizationVariable method*), 54, 57
- `get_min_value()` (*Real_OptimizationVariable method*), 53, 57
- `get_nadir_point()` (*EvolutionaryConvergence method*), 49, 50
- `get_nadir_point_all_steps()` (*EvolutionaryConvergence method*), 49, 50
- `get_name()` (*Attribute_selector method*), 86
- `get_name()` (*Container_attribute_selector method*), 86
- `get_name()` (*FastObjCons method*), 47, 48, 56
- `get_name()` (*InterfaceObjCons method*), 48, 57
- `get_name()` (*on_click_change_symbol method*), 60, 64, 81, 92, 101
- `get_name()` (*on_click_copy_something method*), 60, 63, 81, 92, 101
- `get_name()` (*on_click_extract_pareto method*), 61, 63, 80, 91, 100
- `get_name()` (*on_graph_click_delete method*), 61, 62, 80, 91, 100
- `get_name()` (*on_graph_click_export method*), 61, 62, 80, 91, 100
- `get_name()` (*on_graph_click_remove_trace method*), 61, 63, 81, 92, 101
- `get_name()` (*on_graph_click_showAnim method*), 60, 66, 83, 94, 102
- `get_name()` (*on_graph_click_showInfo method*), 62, 63, 81, 91, 100, 105
- `get_name()` (*Options method*), 32, 45
- `get_nb_objectives()` (*EvolutionaryConvergence method*), 49, 50
- `get_nb_steps()` (*EvolutionaryConvergence method*), 49, 50
- `get_ND_pareto()` (*in module optimeed.core*), 39
- `get_ND_pareto()` (*in module optimeed.core.tools*), 34
- `get_new_index()` (*on_graph_click_showInfo.DataInformationVisualize method*), 62, 63, 80, 91, 100, 105
- `get_number_of_elements()` (*DataAnimation-Trace method*), 84
- `get_number_of_points()` (*Data method*), 27, 40, 108
- `get_object_attrs()` (*in module optimeed.core*), 38
- `get_object_attrs()` (*in module optimeed.core.tools*), 33
- `get_opengl_options()` (*DeviceDrawerInterface method*), 72, 83, 91, 99
- `get_optionValue()` (*Option_class method*), 20, 32, 45
- `get_pareto_convergence()` (*EvolutionaryConvergence method*), 48, 50
- `get_permutations()` (*Data method*), 27, 40, 108
- `get_PhysToMaths()` (*Binary_OptimizationVariable method*), 54, 57
- `get_PhysToMaths()` (*Integer_OptimizationVariable method*), 54, 58
- `get_PhysToMaths()` (*OptimizationVariable method*), 53
- `get_PhysToMaths()` (*Real_OptimizationVariable method*), 53, 57
- `get_plot_data()` (*Data method*), 27, 40, 108
- `get_population_size()` (*EvolutionaryConvergence method*), 49, 51
- `get_reference_device()` (*Parametric_parameter method*), 19, 20
- `get_results()` (*OptiHistoric method*), 55
- `get_styles()` (*in module optimeed.core.ansi2html.style*), 23
- `get_symbol()` (*Data method*), 28, 41, 109
- `get_symbol()` (*TraceVisual method*), 70
- `get_symbolOutline()` (*Data method*), 26, 40, 107
- `get_symbolPens()` (*TraceVisual method*), 71
- `get_symbolsSize()` (*Data method*), 26, 40, 107
- `get_text_to_write()` (*ContextHandler method*), 72
- `get_trace()` (*Graph method*), 28, 42
- `get_trace()` (*GraphVisual method*), 67
- `get_value()` (*Options method*), 32, 45
- `get_values()` (*Parametric_minmax method*), 19, 21
- `get_wgGraphs()` (*in module optimeed.visualize*), 111
- `get_wgGraphs()` (*in module optimeed.visualize.fastPlot*), 95
- `get_wgGraphs()` (*PlotHolders method*), 94, 110
- `get_wgGraphs()` (*WindowHolders method*), 95, 111
- `get_widget()` (*Repr_lines method*), 62, 63, 81, 92, 100
- `get_widget()` (*Repr_opengl method*), 62, 63, 81, 92, 100
- `get_width()` (*Data method*), 27, 40, 108
- `get_x()` (*Data method*), 26, 40, 107
- `get_xlim()` (*Data method*), 27, 41, 108
- `get_xlim()` (*Data method*), 27, 40, 107
- `get_y()` (*Data method*), 27, 40, 107
- `get_y_label()` (*Data method*), 28, 41, 108
- `get_ylim()` (*Data method*), 27, 40, 107

getAmb3() (*MaterialRenderingProperties* method), 73
 getDif3() (*MaterialRenderingProperties* method), 73
 GetEar() (in module *optimeed.visualize.gui.widgets.openGLWidget.TriangulatePolygon*), 74
 getLength() (*MultiList* method), 50
 getLineInfo() (in module *optimeed.core*), 38
 getLineInfo() (in module *optimeed.core.tools*), 33
 getPath_workspace() (in module *optimeed consolidate*), 20
 getPath_workspace() (in module *optimeed.core*), 35, 38
 getPath_workspace() (in module *optimeed.core.tools*), 33
 getShin() (*MaterialRenderingProperties* method), 73
 getSpec3() (*MaterialRenderingProperties* method), 73
 Graph (class in *optimeed.core*), 41
 Graph (class in *optimeed.core.graphs*), 28
 graph_clicked() (*on_click_change_symbol* method), 60, 64, 81, 92, 101
 graph_clicked() (*on_click_copy_something* method), 60, 63, 81, 92, 101
 graph_clicked() (*on_click_extract_pareto* method), 61, 63, 80, 91, 100
 graph_clicked() (*on_graph_click_delete* method), 61, 62, 80, 91, 100
 graph_clicked() (*on_graph_click_export* method), 61, 62, 80, 91, 100
 graph_clicked() (*on_graph_click_remove_trace* method), 61, 63, 81, 92, 100
 graph_clicked() (*on_graph_click_showAnim* method), 60, 65, 83, 94, 102
 graph_clicked() (*on_graph_click_showInfo* method), 62, 63, 81, 91, 100, 105
 Graphs (class in *optimeed.core*), 42
 Graphs (class in *optimeed.core.graphs*), 28
 Graphs (class in *optimeed.visualize*), 109
 GraphVisual (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.graphVisual*), 66
 GREEN (*text_format* attribute), 20, 33, 37, 45
 grid_off() (*GraphVisual* method), 67
 gui_collection_exporter (class in *optimeed.visualize*), 96
 gui_collection_exporter (class in *optimeed.visualize.gui*), 87
 gui_collection_exporter (class in *optimeed.visualize.gui.gui_collection_exporter*), 84
 gui_mainWindow (class in *optimeed.visualize*), 95, 103, 107
 gui_mainWindow (class in *optimeed.visualize.gui*), 87
 gui_mainWindow (class in *optimeed.visualize.gui.gui_mainWindow*), 86
 GuiDataSelector (class in *optimeed.visualize.gui.gui_data_selector*), 86
 guiDataPolygonGraph (class in *optimeed.visualize*), 99, 105, 110
 guiPyqtgraph (class in *optimeed.visualize.gui*), 90
 guiPyqtgraph (class in *optimeed.visualize.gui.widgets*), 83
 guiPyqtgraph (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.smallGui*), 69

H

hide() (*TraceVisual* method), 70
 hide_axes() (*GraphVisual* method), 66
 hide_points() (*TraceVisual* method), 70
 HowToPlotGraph (class in *optimeed.core*), 43
 HowToPlotGraph (class in *optimeed.core.linkDataGraph*), 30
 HowToPlotGraph (class in *optimeed.visualize*), 103
 hvRecursive() (*HyperVolume* method), 49
 HyperVolume (class in *optimeed.optimize.optiAlgorithms.convergence.hypervolume*), 49

I

indentParagraph() (in module *optimeed consolidate*), 20
 indentParagraph() (in module *optimeed.core*), 35, 38
 indentParagraph() (in module *optimeed.core.tools*), 34
 index() (in module *optimeed.core.ansi2html.style*), 23
 index2() (in module *optimeed.core.ansi2html.style*), 23
 initialize() (*ContextHandler* method), 72
 initialize() (*MyTerminationCondition* method), 52
 initialize_output_collection() (*Parametric_analysis* method), 20, 21
 initializeGL() (*widget_openGL* method), 77, 79, 90, 99
 Integer_OptimizationVariable (class in *optimeed.optimize*), 57
 Integer_OptimizationVariable (class in *optimeed.optimize.optiVariable*), 54
 integrate() (in module *optimeed.core*), 38
 integrate() (in module *optimeed.core.tools*), 34
 intensify() (in module *optimeed.core.ansi2html.style*), 23
 InterfaceCharacterization (class in *optimeed.optimize*), 56
 InterfaceCharacterization (class in *optimeed.optimize.characterization*), 46

InterfaceCharacterization (class in optimeed.optimize.characterization.interfaceCharacterization), 46

InterfaceConvergence (class in optimeed.optimize.optiAlgorithms.convergence), 51

InterfaceConvergence (class in optimeed.optimize.optiAlgorithms.convergence.interfaceConvergence), 50

InterfaceDevice (class in optimeed.core), 43

InterfaceDevice (class in optimeed.core.interfaceDevice), 30

InterfaceMathsToPhysics (class in optimeed.optimize), 56

InterfaceMathsToPhysics (class in optimeed.optimize.mathsToPhysics), 47

InterfaceMathsToPhysics (class in optimeed.optimize.mathsToPhysics.interfaceMathsToPhysics), 46

InterfaceObjCons (class in optimeed.optimize), 56

InterfaceObjCons (class in optimeed.optimize.objAndCons), 48

InterfaceObjCons (class in optimeed.optimize.objAndCons.interfaceObjCons), 48

interpolate() (fast_LUT_interpolation method), 35, 39

InTriangle() (in module optimeed.visualize.gui.widgets.openGLWidget.TriangulatePolygon), 74

is_empty() (DataAnimationVisuals method), 64, 82, 85, 88, 93, 97

is_monobj (EvolutionaryConvergence attribute), 48, 50

is_object_selected() (in module optimeed.visualize.gui.gui_data_selector), 86

is_scattered() (Data method), 28, 41, 109

is_slave() (LinkDataGraph method), 31, 44, 102

is_slave() (LinkDataGraph._collection_linker method), 30, 44, 102

IsClockwise() (in module optimeed.visualize.gui.widgets.openGLWidget.TriangulatePolygon), 74

IsConvex() (in module optimeed.visualize.gui.widgets.openGLWidget.TriangulatePolygon), 74

isNonePrintMessage() (in module optimeed.core), 38

isNonePrintMessage() (in module optimeed.core.tools), 33

isOnWindows (in module optimeed.visualize.gui.widgets.graphsVisualWidget.pyqtgraphReDefine), 67

json_to_obj() (in module optimeed.core), 35

json_to_obj() (in module optimeed.core.myjson), 32

json_to_obj_safe() (in module optimeed.core), 35

json_to_obj_safe() (in module optimeed.core.myjson), 32

keyboard_push_action() (DeviceDrawerInterface method), 72, 83, 91, 99

keyboardPushAction() (ContextHandler method), 72

keyboardReleaseAction() (ContextHandler method), 72

keyPressEvent() (gui_mainWindow method), 87, 96, 103, 107

keyPressEvent() (widget_graphs_visual method), 75, 78, 89, 98, 104, 106

keyPressEvent() (widget_openGL method), 77, 80, 90, 99

KWARGS_OPTIHISTO (Optimizer attribute), 55, 58

L

launch_optimization() (OptimizationDisplayer method), 94, 105

level() (in module optimeed.core.ansi2html.style), 23

link_axes() (widget_graphs_visual method), 75, 78, 89, 97, 104, 106

link_collection_to_graph_collection() (LinkDataGraph method), 31, 44, 102

LinkDataGraph (class in optimeed.core), 44

LinkDataGraph (class in optimeed.core.linkDataGraph), 30

LinkDataGraph (class in optimeed.visualize), 102

LinkDataGraph._collection_linker (class in optimeed.core), 44

LinkDataGraph._collection_linker (class in optimeed.core.linkDataGraph), 30

LinkDataGraph._collection_linker (class in optimeed.visualize), 102

linkify() (in module optimeed.core.ansi2html.converter), 22

linkXToGraph() (GraphVisual method), 67

linspace() (in module optimeed.core), 39

linspace() (in module optimeed.core.tools), 34

ListDataStruct (class in optimeed.core), 36

ListDataStruct (class in optimeed.core.collection), 25

M

main() (in module optimeed.core.ansi2html.converter), 23

manage_list() (in module optimeed.visualize.gui.gui_data_selector), 86

map_index() (*DataAnimationTrace* method), 84
 map_vt100_box_code() (in module *optimeed.core.ansi2html.converter*), 22
 MaterialRenderingProperties (class in *optimeed.visualize.gui.widgets.openGLWidget.Materials_visualize*), 72
 MathsToPhysics (class in *optimeed.optimize*), 56
 MathsToPhysics (class in *optimeed.optimize.mathsToPhysics*), 47
 MathsToPhysics (class in *optimeed.optimize.mathsToPhysics.mathsToPhysics*), 47
 merge() (*Graphs* method), 29, 43, 110
 merge() (*ListDataStruct* method), 25, 37
 meshPolygon() (in module *optimeed.visualize.gui.widgets.openGLWidget.TriangulatePolygon*), 74
 minimumSizeHint() (*widget_openGL* method), 77, 79, 90, 99
 MODE_LIGHT (in module *optimeed.visualize.gui.widgets.openGLWidget.ContextHandler*), 71
 MODE_ROTATION (in module *optimeed.visualize.gui.widgets.openGLWidget.ContextHandler*), 71
 MODE_ZOOM (in module *optimeed.visualize.gui.widgets.openGLWidget.ContextHandler*), 71
 modify_paintElems() (*TraceVisual._ModifiedPaintElem* method), 69
 MODULE_TAG (in module *optimeed.core.myjson*), 31
 mouseClicAction() (*ContextHandler* method), 72
 mouseMotionAction() (*ContextHandler* method), 72
 mouseMoveEvent() (*widget_openGL* method), 77, 80, 90, 99
 mousePressEvent() (*widget_openGL* method), 77, 80, 90, 99
 mouseWheelAction() (*ContextHandler* method), 72
 MultiList (class in *optimeed.optimize.optiAlgorithms.convergence.hypervolume*), 49
 MultiList.Node (class in *optimeed.optimize.optiAlgorithms.convergence.hypervolume*), 49
 MultiObjective_GA (class in *optimeed.optimize*), 57
 MultiObjective_GA (class in *optimeed.optimize.optiAlgorithms*), 53
 MultiObjective_GA (class in *optimeed.optimize.optiAlgorithms.multiObjective_GA*), 52
 my_fourier() (in module *optimeed.core*), 39
 my_fourier() (in module *optimeed.core.tools*), 34
 myAxis (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.pyqtgraphRedefined*), 69
 MyConvergence (class in *optimeed.optimize.optiAlgorithms.multiObjective_GA*), 51
 MyGenerator (class in *optimeed.optimize.optiAlgorithms.multiObjective_GA*), 52
 myGraphicsLayout (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.pyqtgraphRedefined*), 68
 myGraphicsLayoutWidget (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.pyqtgraphRedefined*), 67
 myItemSample (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.pyqtgraphRedefined*), 68
 myLabelItem (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.pyqtgraphRedefined*), 69
 myLabelWidget (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.pyqtgraphRedefined*), 68
 myEvaluator (class in *optimeed.optimize.optiAlgorithms.multiObjective_GA*), 52
 myProcessEvaluator (class in *optimeed.optimize.optiAlgorithms.multiObjective_GA*), 52
 MyProblem (class in *optimeed.optimize.optiAlgorithms.multiObjective_GA*), 52
 MyTerminationCondition (class in *optimeed.optimize.optiAlgorithms.multiObjective_GA*), 52
 MyText (class in *optimeed.visualize.gui.widgets.openGLWidget.ContextHandler*), 72
 myWindows (in module *optimeed.visualize*), 111
 myWindows (in module *optimeed.visualize.fastPlot*), 95

N

new_figure() (*WindowHolders* method), 95, 110
 new_plot() (in module *optimeed.visualize*), 111
 new_plot() (in module *optimeed.visualize.fastPlot*), 95
 new_plot() (*PlotHolders* method), 94, 110
 new_plot() (*WindowHolders* method), 95, 111
 next_frame() (*DataAnimationVisuals* method), 64, 82, 85, 88, 93, 97
 NLOpt_Algorithm (class in *optimeed.optimize.optiAlgorithms.NLOpt_Algorithm*), 51

normalize() (in module optimeed.visualize.gui.widgets.openGLWidget.QUATERNIONS), 74

NUMBER_OF_CORES (MultiObjective_GA attribute), 52, 53, 57

NUMBER_OF_CORES (Parametric_analysis attribute), 19, 21

NUMBER_OF_MODES (in module optimeed.visualize.gui.widgets.openGLWidget.ContextHandler), 71

O

obj_to_json() (in module optimeed.core), 35

obj_to_json() (in module optimeed.core.myjson), 32

objectives (OptiHistoric._pointData attribute), 54

objectives_per_step (EvolutionaryConvergence attribute), 48, 50

on_click() (widget_graphs_visual method), 75, 78, 88, 97, 103, 106

on_click_change_symbol (class in optimeed.visualize), 101

on_click_change_symbol (class in optimeed.visualize.gui), 92

on_click_change_symbol (class in optimeed.visualize.gui.widgets), 81

on_click_change_symbol (class in optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick), 63

on_click_change_symbol (class in optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick.on_click_change_symbol), 60

on_click_copy_something (class in optimeed.visualize), 101

on_click_copy_something (class in optimeed.visualize.gui), 92

on_click_copy_something (class in optimeed.visualize.gui.widgets), 81

on_click_copy_something (class in optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick), 63

on_click_copy_something (class in optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick.on_click_copy_something), 60

on_click_extract_pareto (class in optimeed.visualize), 100

on_click_extract_pareto (class in optimeed.visualize.gui), 91

on_click_extract_pareto (class in optimeed.visualize.gui.widgets), 80

on_click_extract_pareto (class in optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick), 63

on_click_extract_pareto (class in optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick.on_click_extract_pareto), 63

on_graph_click_delete (class in optimeed.visualize), 99

on_graph_click_delete (class in optimeed.visualize.gui), 91

on_graph_click_delete (class in optimeed.visualize.gui.widgets), 80

on_graph_click_delete (class in optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick), 62

on_graph_click_delete (class in optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick.on_click_delete), 60

on_graph_click_export (class in optimeed.visualize), 100

on_graph_click_export (class in optimeed.visualize.gui), 91

on_graph_click_export (class in optimeed.visualize.gui.widgets), 80

on_graph_click_export (class in optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick), 62

on_graph_click_export (class in optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick.on_click_export), 61

on_graph_click_interface (class in optimeed.visualize), 101

on_graph_click_interface (class in optimeed.visualize.gui), 92

on_graph_click_interface (class in optimeed.visualize.gui.widgets), 81

on_graph_click_interface (class in optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick), 64

on_graph_click_interface (class in optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick.on_click_interface), 74

on_graph_click_remove_trace (class in optimeed.visualize), 100

on_graph_click_remove_trace (class in optimeed.visualize.gui), 92

on_graph_click_remove_trace (class in optimeed.visualize.gui.widgets), 81

on_graph_click_remove_trace (class in optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick), 63

on_graph_click_remove_trace (class in optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick.on_click_remove_trace), 61

on_graph_click_showAnim (class in optimeed.visualize), 101

on_graph_click_showAnim (class in optimeed.visualize.gui), 93

on_graph_click_showAnim (class in optimeed.visualize.gui.widgets), 81

on_graph_click_showAnim (class in optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick), 63

on_graph_click_showAnim (class in optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick.on_click_showAnim), 63

meed.visualize.gui.widgets), 83
 on_graph_click_showAnim (class in optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick), 65
 on_graph_click_showAnim (class in optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClickOnClick), 60
 on_graph_click_showInfo (class in optimeed.visualize), 100, 104
 on_graph_click_showInfo (class in optimeed.visualize.gui), 91
 on_graph_click_showInfo (class in optimeed.visualize.gui.widgets), 80
 on_graph_click_showInfo (class in optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick), 63
 on_graph_click_showInfo (class in optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClickOnClick), 62
 on_graph_click_showInfo.DataInformationVisuals (class in optimeed.visualize), 100, 104
 on_graph_click_showInfo.DataInformationVisuals (module), 47
 on_graph_click_showInfo.DataInformationVisuals (class in optimeed.visualize.gui), 91
 on_graph_click_showInfo.DataInformationVisuals (module), 47
 on_graph_click_showInfo.DataInformationVisuals (class in optimeed.visualize.gui.widgets), 80
 on_graph_click_showInfo.DataInformationVisuals (class in optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClick), 63
 on_graph_click_showInfo.DataInformationVisuals (module), 48
 on_graph_click_showInfo.DataInformationVisuals (class in optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionOnClickOnClick), 62
 on_update_signal() (widget_line_drawer method), 65, 76, 79, 82, 90, 98
 OPTI_ALGORITHM (MultiObjective_GA attribute), 52, 53, 57
 OptiHistoric (class in optimeed.optimize.optimizer), 54
 OptiHistoric._pointData (class in optimeed.optimize.optimizer), 54
 optimeed (module), 19
 optimeed consolidate (module), 19
 optimeed consolidate parametric_analysis (module), 19
 optimeed.core (module), 21
 optimeed.core ansi2html (module), 21
 optimeed.core ansi2html.converter (module), 21
 optimeed.core ansi2html.style (module), 23
 optimeed.core ansi2html.util (module), 23
 optimeed.core.collection (module), 24
 optimeed.core.color_palette (module), 26
 optimeed.core.commonImport (module), 26
 optimeed.core.graphs (module), 26
 optimeed.core.interfaceDevice (module), 30
 optimeed.core.linkDataGraph (module), 30
 optimeed.core.myjson (module), 31
 optimeed.core.options (module), 32
 optimeed.core.on_click (module), 33
 optimeed.optimize (module), 45
 optimeed.optimize.characterization (module), 46
 optimeed.optimize.characterization.characterization (module), 46
 optimeed.optimize.characterization.interfaceCharacterization (module), 46
 optimeed.optimize.mathsToPhysics (module), 46
 optimeed.optimize.mathsToPhysics.interfaceMathsToPhysics (module), 46
 optimeed.optimize.mathsToPhysics.mathsToPhysics (module), 47
 optimeed.optimize.objAndCons (module), 47
 optimeed.optimize.objAndCons.fastObjCons (module), 47
 optimeed.optimize.objAndCons.interfaceObjCons (module), 47
 optimeed.optimize.optiAlgorithms (module), 48
 optimeed.optimize.optiAlgorithms.algorithmInterface (module), 48
 optimeed.optimize.optiAlgorithms.convergence (module), 49
 optimeed.optimize.optiAlgorithms.convergence.evolution (module), 49
 optimeed.optimize.optiAlgorithms.convergence.hyperparameter (module), 49
 optimeed.optimize.optiAlgorithms.convergence.interfaceConvergence (module), 50
 optimeed.optimize.optiAlgorithms.multiObjective_GA (module), 51
 optimeed.optimize.optiAlgorithms.NLOpt_Algorithm (module), 51
 optimeed.optimize.optimizer (module), 54
 optimeed.optimize.optiVariable (module), 53
 optimeed.visualize (module), 59
 optimeed.visualize.displayOptimization (module), 94
 optimeed.visualize.fastPlot (module), 94
 optimeed.visualize.gui (module), 59
 optimeed.visualize.gui.gui_collection_exporter (module), 84
 optimeed.visualize.gui.gui_data_animation (module), 84
 optimeed.visualize.gui.gui_data_selector (module), 85

POPULATION_SIZE (*NLOpt_Algorithm* attribute), 51
 prepare() (*Ansi2HTMLConverter* method), 23, 24
 preProcess() (*HyperVolume* method), 49
 printIfShown() (in module *optimeed.core*), 36, 38, 39
 printIfShown() (in module *optimeed.core.tools*), 33
 produce_headers() (*Ansi2HTMLConverter* method), 23, 24
 PURPLE (*text_format* attribute), 20, 33, 37, 45

Q

q_conjugate() (in module *optimeed.visualize.gui.widgets.openGLWidget.quaternions*), 74
 q_mult() (in module *optimeed.visualize.gui.widgets.openGLWidget.quaternions*), 74
 q_to_axisangle() (in module *optimeed.visualize.gui.widgets.openGLWidget.quaternions*), 74
 q_to_mat4() (in module *optimeed.visualize.gui.widgets.openGLWidget.quaternions*), 74
 quicksort() (in module *optimeed.core*), 38
 quicksort() (in module *optimeed.core.tools*), 34
 qv_mult() (in module *optimeed.visualize.gui.widgets.openGLWidget.quaternions*), 74

R

read_to_unicode() (in module *optimeed.core.ansi2html.util*), 23
 Real_OptimizationVariable (class in *optimeed.optimize*), 57
 Real_OptimizationVariable (class in *optimeed.optimize.optiVariable*), 53
 RED (*text_format* attribute), 20, 33, 37, 45
 Red_material (in module *optimeed.visualize.gui.widgets.openGLWidget.Materials_visual*), 73
 redraw() (*ContextHandler* method), 72
 reformatXYtoList() (in module *optimeed.visualize.gui.widgets.openGLWidget.TriangulatePolygon*), 74
 refreshTraceList() (*guiPyqtgraph* method), 69, 83, 90, 99, 105, 110
 reinsert() (*MultiList* method), 50
 remove() (*MultiList* method), 50
 remove_element_from_graph() (*LinkData-Graph* method), 31, 44, 102
 remove_elements_from_trace() (*LinkData-Graph* method), 31, 44, 102
 remove_feature() (*GraphVisual* method), 66
 remove_graph() (*Graphs* method), 29, 43, 110

remove_trace() (*Graph* method), 28, 42
 remove_trace() (*Graphs* method), 29, 42, 109
 remove_trace() (*LinkDataGraph* method), 31, 44, 102
 Repr_lines (class in *optimeed.visualize*), 100
 Repr_lines (class in *optimeed.visualize.gui*), 92
 Repr_lines (class in *optimeed.visualize.gui.widgets*), 81

Repr_lines (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionO*), 63

Repr_lines (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionO*), 62

Repr_opengl (class in *optimeed.visualize*), 100
 Repr_opengl (class in *optimeed.visualize.gui*), 92

Repr_opengl (class in *optimeed.visualize.gui.widgets*), 81

Repr_opengl (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionO*), 63

Repr_opengl (class in *optimeed.visualize.gui.widgets.graphsVisualWidget.examplesActionO*), 62

reset() (*_State* method), 22
 reset() (*AlgorithmInterface* method), 51
 reset() (*Graphs* method), 29, 43, 110
 reset() (*gui_collection_exporter* method), 84, 87, 96
 reset() (*on_graph_click_delete* method), 61, 62, 80, 91, 100

reset() (*PlotHolders* method), 95, 110
 reset() (*TraceVisual._ModifiedPaintElem* method), 70
 reset_all() (*DataAnimationVisuals* method), 64, 82, 85, 88, 93, 96

reset_all_brushes() (*TraceVisual* method), 71
 reset_all_symbolPens() (*TraceVisual* method), 71

reset_brush() (*TraceVisual* method), 71
 reset_graph() (*on_graph_click_export* method), 61, 62, 80, 91, 100

reset_paintElem() (*TraceVisual._ModifiedPaintElem* method), 69

reset_symbol() (*TraceVisual* method), 71
 reset_symbolPen() (*TraceVisual* method), 71

resizeEvent() (*myAxis* method), 69
 resizeGL() (*widget_openGL* method), 77, 79, 90, 99
 resizeWindowAction() (*ContextHandler* method), 72

rgetattr() (in module *optimeed consolidate*), 20
 rgetattr() (in module *optimeed.core*), 35, 38
 rgetattr() (in module *optimeed.core.tools*), 34
 rsetattr() (in module *optimeed consolidate*), 20
 rsetattr() (in module *optimeed.core*), 38
 rsetattr() (in module *optimeed.core.tools*), 34

Rule (class in *optimeed.core.ansi2html.style*), 23
 run() (*DataAnimationVisuals* method), 64, 82, 85, 88, 93, 97
 run() (*gui_mainWindow* method), 86, 87, 96, 103, 107
 run() (*GuiDataSelector* method), 86
 run() (*Parametric_analysis* method), 19, 21
 run_optimization() (*Optimizer* method), 55, 58

S

save() (*AutosaveStruct* method), 25, 36
 save() (*ListDataStruct* method), 25, 36
 save() (*OptiHistoric* method), 55
 SaveableObject (class in *optimeed.core*), 35
 SaveableObject (class in *optimeed.core.myjson*), 31
 SCHEME (in module *optimeed.core.ansi2html.style*), 23
 scrollable_widget_text (class in *optimeed.visualize.gui.widgets.widget_text*), 77
 set_actionOnClick() (*widget_graphs_visual* method), 75, 78, 89, 98, 104, 106
 set_actionOnClose() (*gui_mainWindow* method), 86, 87, 95, 103, 107
 set_actionsOnClick() (*OptimizationDisplayer* method), 94, 105
 set_all_options() (*Option_class* method), 20, 32, 45
 set_article_template() (*widget_graphs_visual* method), 76, 79, 89, 98, 104, 106
 set_attribute_data() (*ListDataStruct* method), 25, 37
 set_attribute_equation() (*ListDataStruct* method), 25, 37
 set_attribute_selectors() (*Container_attribute_selector* method), 86
 set_brush() (*TraceVisual* method), 70
 set_brushes() (*TraceVisual* method), 71
 set_collection() (*gui_collection_exporter* method), 84, 87, 96
 set_color() (*TraceVisual* method), 70
 set_color_palette() (*GraphVisual* method), 66
 set_convergence() (*OptiHistoric* method), 55
 set_curr_brush() (*DataAnimationTrace* method), 84
 set_currFigure() (*WindowHolders* method), 95, 110
 set_data() (*Data* method), 26, 40, 107
 set_data() (*ListDataStruct* method), 25, 36
 set_data_at_index() (*ListDataStruct* method), 25, 36
 set_device() (*PipeOptimization* method), 54
 set_deviceDrawer() (*ContextHandler* method), 72
 set_deviceDrawer() (*widget_openGL* method), 77, 79, 90, 99
 set_deviceToDraw() (*ContextHandler* method), 72

set_deviceToDraw() (*widget_openGL* method), 77, 79, 90, 99
 set_evaluationFunction() (*MultiObjective_GA* method), 52, 53, 57
 set_evaluationFunction() (*NLOpt_Algorithm* method), 51
 set_filename() (*AutosaveStruct* method), 24, 36
 set_font() (*myLegend* method), 68
 set_fontLabel() (*GraphVisual* method), 66
 set_fontLegend() (*GraphVisual* method), 66
 set_fontTicks() (*GraphVisual* method), 66
 set_graph_disposition() (*myGraphicsLayout* method), 68
 set_graph_disposition() (*widget_graphs_visual* method), 75, 78, 88, 97, 103, 105
 set_graph_properties() (*GraphVisual* method), 67
 set_historic() (*PipeOptimization* method), 54
 set_idle_brush() (*DataAnimationTrace* method), 84
 set_indices_points_to_plot() (*Data* method), 28, 41, 109
 set_info() (*DataStruct_Interface* method), 24, 36
 set_info() (*gui_collection_exporter* method), 84, 87, 96
 set_info() (*OptiHistoric* method), 55
 set_label_pos() (*GraphVisual* method), 66
 set_label_pos() (*myAxis* method), 69
 set_legend() (*GraphVisual* method), 67
 set_lims() (*GraphVisual* method), 67
 set_lines() (*widget_line_drawer* method), 65, 76, 79, 82, 90, 98
 set_max_opti_time() (*Optimizer* method), 55, 58
 set_maxtime() (*MultiObjective_GA* method), 52, 53, 57
 set_maxtime() (*NLOpt_Algorithm* method), 51
 set_number_ticks() (*myAxis* method), 69
 set_numberTicks() (*GraphVisual* method), 66
 set_offset() (*myItemSample* method), 68
 set_offset_sample() (*myLegend* method), 68
 set_optimizer() (*Optimizer* method), 55, 58
 set_option() (*Options* method), 32, 45
 set_optionValue() (*Option_class* method), 20, 32, 45
 set_permutations() (*Data* method), 27, 41, 108
 set_points_at_step() (*EvolutionaryConvergence* method), 48, 50
 set_pop_size() (*ConvergenceManager* method), 51
 set_position() (*myLegend* method), 68
 set_refreshTime() (*DataAnimationVisuals* method), 64, 82, 85, 88, 93, 97
 set_results() (*OptiHistoric* method), 55
 set_same_master() (*LinkData-*

- Graph_collection_linker* method), 30, 44, 102
- set_self()* (*Options* method), 32, 45
- set_space_sample_label()* (*myLegend* method), 68
- set_specialButtonsMapping()* (*ContextHandler* method), 72
- set_symbol()* (*TraceVisual* method), 70
- set_symbolPen()* (*TraceVisual* method), 71
- set_symbolPens()* (*TraceVisual* method), 71
- set_text()* (*scrollable_widget_text* method), 77
- set_text()* (*widget_text* method), 64, 77, 80, 82, 90, 99
- set_title()* (*GraphVisual* method), 67
- set_title()* (in module *optimeed.visualize*), 111
- set_title()* (in module *optimeed.visualize.fastPlot*), 95
- set_title()* (*PlotHolders* method), 95, 110
- set_title()* (*widget_graphs_visual* method), 76, 78, 89, 98, 104, 106
- set_title()* (*WindowHolders* method), 95, 110
- set_width_cell()* (*myItemSample* method), 68
- set_width_cell_sample()* (*myLegend* method), 68
- setText()* (*myLabelItem* method), 69
- shouldTerminate()* (*MyTerminationCondition* method), 52
- show()* (in module *optimeed.visualize*), 111
- show()* (in module *optimeed.visualize.fastPlot*), 95
- show()* (*TraceVisual* method), 70
- show()* (*WindowHolders* method), 95, 111
- show_all()* (*DataAnimationTrace* method), 84
- show_all()* (*DataAnimationVisuals* method), 64, 82, 85, 88, 93, 97
- SHOW_CURRENT* (in module *optimeed.core*), 39, 43
- SHOW_CURRENT* (in module *optimeed.core.commonImport*), 26
- SHOW_DEBUG* (in module *optimeed.core*), 39, 43
- SHOW_DEBUG* (in module *optimeed.core.commonImport*), 26
- SHOW_ERROR* (in module *optimeed.core*), 39, 43
- SHOW_ERROR* (in module *optimeed.core.commonImport*), 26
- SHOW_INFO* (in module *optimeed.core*), 39, 43
- SHOW_INFO* (in module *optimeed.core.commonImport*), 26
- SHOW_WARNING* (in module *optimeed.core*), 36, 39, 43
- SHOW_WARNING* (in module *optimeed.core.commonImport*), 26
- showEvent()* (*widget_menuButton* method), 76, 79, 90, 99
- signal_graph_changed* (*widget_graphs_visual* attribute), 75, 78, 88, 97, 103, 105
- signal_has_exported* (*gui_collection_exporter* attribute), 84, 87, 96
- signal_has_reset* (*gui_collection_exporter* attribute), 84, 87, 96
- signal_must_update* (*TraceVisual* attribute), 70
- signal_must_update* (*widget_graphs_visual* attribute), 75, 77, 88, 97, 103, 105
- signal_must_update* (*widget_line_drawer* attribute), 65, 76, 79, 82, 89, 98
- signal_optimization_over* (*OptimizationDisplayer* attribute), 94, 105
- signal_show_UI* (*Worker* attribute), 94, 105
- Silver_material* (in module *optimeed.visualize.gui.widgets.openGLWidget.Materials_visual*), 73
- sizeHint()* (*widget_openGL* method), 77, 79, 90, 99
- slider_handler()* (*DataAnimationVisuals* method), 64, 82, 85, 88, 93, 97
- SLIDER_MAXIMUM_VALUE* (*DataAnimationVisuals* attribute), 64, 81, 85, 87, 92, 96
- SLIDER_MINIMUM_VALUE* (*DataAnimationVisuals* attribute), 64, 81, 85, 87, 92, 96
- software_version()* (in module *optimeed.core*), 37
- software_version()* (in module *optimeed.core.tools*), 33
- sortByDimension()* (*HyperVolume* method), 49
- sparse_subset()* (in module *optimeed.core*), 38
- sparse_subset()* (in module *optimeed.core.tools*), 34
- SpecialButtonsMapping* (class in *optimeed.visualize.gui.widgets.openGLWidget.ContextHandler*), 72
- start_autosave()* (*AutosaveStruct* method), 24, 36
- start_qt_mainloop()* (in module *optimeed.visualize*), 96, 107
- start_qt_mainloop()* (in module *optimeed.visualize.gui*), 87
- start_qt_mainloop()* (in module *optimeed.visualize.gui.gui_mainWindow*), 86
- Steel_material* (in module *optimeed.visualize.gui.widgets.openGLWidget.Materials_visual*), 73
- stop_autosave()* (*AutosaveStruct* method), 24, 36
- stop_qt_mainloop()* (in module *optimeed.visualize*), 96, 107
- stop_qt_mainloop()* (in module *optimeed.visualize.gui*), 87
- stop_qt_mainloop()* (in module *optimeed.visualize.gui.gui_mainWindow*), 86
- str_all_attr()* (in module *optimeed.core*), 39
- str_all_attr()* (in module *optimeed.core.tools*), 34
- symbol_isfilled()* (*Data* method), 26, 40, 107

T

text_format (class in *optimeed consolidate*), 20
 text_format (class in *optimeed core*), 37, 45
 text_format (class in *optimeed core tools*), 33
 theActionOnUpdate (*GuiDataSelector* attribute), 86
 time (*OptiHistoric._pointData* attribute), 54
 to_css_classes () (*_State* method), 22
 toggle () (*TraceVisual* method), 70
 TraceVisual (class in *optimeed visualize gui widgets graphs VisualWidget.traceVisual*), 69
 TraceVisual._ModifiedPaintElem (class in *optimeed visualize gui widgets graphs VisualWidget.traceVisual*), 69
 truncate () (in module *optimeed core*), 39
 truncate () (in module *optimeed core tools*), 34

U

UNDERLINE (*text_format* attribute), 20, 33, 37, 45
 universalPath () (in module *optimeed core*), 38
 universalPath () (in module *optimeed core tools*), 33
 update () (*GraphVisual* method), 67
 update_graphs () (*LinkDataGraph* method), 31, 44, 102
 update_graphs () (*widget_graphs_visual* method), 75, 78, 89, 97, 103, 106
 update_widget_w_animation () (*DataAnimationLines* method), 59, 65, 83, 93, 101
 update_widget_w_animation () (*DataAnimationOpenGL* method), 59, 65, 82, 93, 101
 update_widget_w_animation () (*DataAnimationOpenGLwText* method), 59, 65, 83, 93, 101
 update_widget_w_animation () (*DataAnimationVisualswText* method), 60, 65, 83, 93, 101
 updateChildren () (*Graphs* method), 28, 42, 109
 updateSize () (*myLegend* method), 68
 updateTrace () (*TraceVisual* method), 70
 useOpenGL () (*myGraphicsLayoutWidget* method), 67

V

VERSION (in module *optimeed*), 111
 VT100_BOX_CODES (in module *optimeed core ansi2html.converter*), 22

W

wheelEvent () (*widget_openGL* method), 77, 80, 90, 99
 WHITE (*text_format* attribute), 20, 33, 37, 45
 widget_graphs_visual (class in *optimeed visualize*), 97, 103, 105

widget_graphs_visual (class in *optimeed visualize gui*), 88
 widget_graphs_visual (class in *optimeed visualize gui widgets*), 77
 widget_graphs_visual (class in *optimeed visualize gui widgets widget_graphs_visual*), 74
 widget_line_drawer (class in *optimeed visualize*), 98
 widget_line_drawer (class in *optimeed visualize gui*), 89
 widget_line_drawer (class in *optimeed visualize gui widgets*), 79, 82
 widget_line_drawer (class in *optimeed visualize gui widgets graphs VisualWidget.examplesActionO*), 64
 widget_line_drawer (class in *optimeed visualize gui widgets widget_line_drawer*), 76
 widget_menuButton (class in *optimeed visualize*), 98
 widget_menuButton (class in *optimeed visualize gui*), 90
 widget_menuButton (class in *optimeed visualize gui widgets*), 79
 widget_menuButton (class in *optimeed visualize gui widgets widget_menuButton*), 76
 widget_openGL (class in *optimeed visualize*), 99
 widget_openGL (class in *optimeed visualize gui*), 90
 widget_openGL (class in *optimeed visualize gui widgets*), 79
 widget_openGL (class in *optimeed visualize gui widgets widget_openGL*), 77
 widget_text (class in *optimeed visualize*), 99
 widget_text (class in *optimeed visualize gui*), 90
 widget_text (class in *optimeed visualize gui widgets*), 80, 82
 widget_text (class in *optimeed visualize gui widgets graphs VisualWidget.examplesActionO*), 64
 widget_text (class in *optimeed visualize gui widgets widget_text*), 77
 WindowHolders (class in *optimeed visualize*), 110
 WindowHolders (class in *optimeed visualize fastPlot*), 95
 Worker (class in *optimeed visualize*), 105
 Worker (class in *optimeed visualize displayOptimization*), 94

Y

YELLOW (*text_format* attribute), 20, 33, 37, 45

Yellow_Emerald_material (in module *optimeed.visualize.gui.widgets.openGLWidget.Materials_visual*),
[73](#)