

Projet\_P5C006

Generated by Doxygen 1.8.14



# Contents

<b>1</b>	<b>Namespace Index</b>	<b>1</b>
1.1	Namespace List . . . . .	1
<b>2</b>	<b>Hierarchical Index</b>	<b>3</b>
2.1	Class Hierarchy . . . . .	3
<b>3</b>	<b>Class Index</b>	<b>5</b>
3.1	Class List . . . . .	5
<b>4</b>	<b>File Index</b>	<b>7</b>
4.1	File List . . . . .	7
<b>5</b>	<b>Namespace Documentation</b>	<b>9</b>
5.1	client Namespace Reference . . . . .	9
5.1.1	Function Documentation . . . . .	9
5.1.1.1	adjacency_matrix() . . . . .	9
5.1.1.2	automatic_partitioning() . . . . .	10
5.1.1.3	casdetude() . . . . .	10
5.1.1.4	casdetude_dinardo() . . . . .	10
5.1.1.5	paesi_example() . . . . .	10
5.1.1.6	render_vtk() . . . . .	10
5.1.1.7	vesuvio_example() . . . . .	11
5.1.2	Variable Documentation . . . . .	11
5.1.2.1	last_slash . . . . .	11
5.1.2.2	PROJECT_PATH . . . . .	11

5.2	graphIO Namespace Reference . . . . .	11
5.3	hardy_cross Namespace Reference . . . . .	11
5.3.1	Function Documentation . . . . .	12
5.3.1.1	add_string_from_list() . . . . .	12
5.3.1.2	diameter() . . . . .	12
5.3.1.3	diameter_from_available() . . . . .	12
5.3.1.4	flow_correction_dq() . . . . .	12
5.3.1.5	j_loss_10atm() . . . . .	13
5.3.1.6	velocity() . . . . .	13
5.4	kpi_calculator Namespace Reference . . . . .	13
5.5	router3 Namespace Reference . . . . .	13
<b>6</b>	<b>Class Documentation</b>	<b>15</b>
6.1	graphIO.display_graph Class Reference . . . . .	15
6.1.1	Detailed Description . . . . .	15
6.1.2	Constructor & Destructor Documentation . . . . .	15
6.1.2.1	__init__() . . . . .	15
6.1.3	Member Function Documentation . . . . .	16
6.1.3.1	coord2D() . . . . .	16
6.1.3.2	display_mesh() . . . . .	16
6.1.3.3	display_path() . . . . .	16
6.1.3.4	distance2D() . . . . .	16
6.1.3.5	distance3D() . . . . .	17
6.1.4	Member Data Documentation . . . . .	17
6.1.4.1	graph . . . . .	17
6.2	graphIO.graph_reader Class Reference . . . . .	17
6.2.1	Detailed Description . . . . .	17
6.2.2	Constructor & Destructor Documentation . . . . .	17
6.2.2.1	__init__() . . . . .	18
6.2.3	Member Function Documentation . . . . .	18
6.2.3.1	avg() . . . . .	18

6.2.3.2	read_adjacency()	18
6.2.3.3	read_epanet()	18
6.2.3.4	read_shp()	18
6.2.3.5	read_shp_bilding()	19
6.2.3.6	row_chuncker()	19
6.2.4	Member Data Documentation	19
6.2.4.1	graph	19
6.3	graphIO.graph_writer Class Reference	20
6.4	hardy_cross.HardyCross Class Reference	20
6.4.1	Constructor & Destructor Documentation	20
6.4.1.1	__init__()	20
6.4.2	Member Function Documentation	21
6.4.2.1	compute_pipe_diameter_of_each_loop()	21
6.4.2.2	compute_velocities_of_each_loop()	21
6.4.2.3	locate_common_loops()	21
6.4.2.4	run_hc()	21
6.4.2.5	save_flows_to_file()	21
6.4.2.6	sort_edge_names()	21
6.4.3	Member Data Documentation	22
6.4.3.1	common_loops	22
6.4.3.2	delta_Qs	22
6.4.3.3	loops	22
6.4.3.4	max_velocity	22
6.4.3.5	min_velocity	22
6.4.3.6	new_D	22
6.4.3.7	runs	22
6.4.3.8	smallest_flow_rate	23
6.4.3.9	threshold	23
6.4.3.10	velocities	23
6.5	kpi_calculator.kpi_calculator Class Reference	23

6.5.1	Detailed Description . . . . .	24
6.5.2	Constructor & Destructor Documentation . . . . .	24
6.5.2.1	__init__() . . . . .	24
6.5.3	Member Function Documentation . . . . .	24
6.5.3.1	print_kpi() . . . . .	24
6.5.4	Member Data Documentation . . . . .	24
6.5.4.1	available_power [1/2] . . . . .	24
6.5.4.2	available_power [2/2] . . . . .	24
6.5.4.3	dissipated_power [1/2] . . . . .	25
6.5.4.4	dissipated_power [2/2] . . . . .	25
6.5.4.5	MAX [1/2] . . . . .	25
6.5.4.6	MAX [2/2] . . . . .	25
6.5.4.7	MEAN . . . . .	25
6.5.4.8	MIN [1/2] . . . . .	25
6.5.4.9	MIN [2/2] . . . . .	25
6.5.4.10	nodes_power . . . . .	25
6.5.4.11	SQM . . . . .	26
6.5.4.12	wdn . . . . .	26
6.6	router3.Router Class Reference . . . . .	26
6.6.1	Constructor & Destructor Documentation . . . . .	27
6.6.1.1	__init__() . . . . .	27
6.6.2	Member Function Documentation . . . . .	27
6.6.2.1	add_node_unique() . . . . .	27
6.6.2.2	cluster() . . . . .	27
6.6.2.3	complete_graph() . . . . .	28
6.6.2.4	compute_source_matrix() . . . . .	28
6.6.2.5	design_aqueduct() . . . . .	28
6.6.2.6	design_minimal_aqueduct() . . . . .	28
6.6.2.7	distance() . . . . .	29
6.6.2.8	graphToEdgeMatrix() . . . . .	29

6.6.2.9	is_source_sink()	29
6.6.2.10	louvain_clustering()	29
6.6.2.11	mesh_graph()	30
6.6.2.12	path_length()	30
6.6.2.13	read_vtk()	30
6.6.2.14	routing()	30
6.6.2.15	shortest_path()	31
6.6.2.16	solve()	31
6.6.2.17	write2epanet()	31
6.6.2.18	write2shp()	31
6.6.2.19	write2vtk()	32
6.6.3	Member Data Documentation	32
6.6.3.1	acqueduct	32
6.6.3.2	acqueduct_1level	32
6.6.3.3	acqueduct_2level	32
6.6.3.4	CLASS_AUTHOR	32
6.6.3.5	CLASS_NAME	32
6.6.3.6	graph	32
6.6.3.7	sinksources_graph	32
<b>7</b>	<b>File Documentation</b>	<b>33</b>
7.1	source/client.py File Reference	33
7.2	source/graphIO.py File Reference	33
7.3	source/hardy_cross.py File Reference	34
7.4	source/kpi_calculator.py File Reference	34
7.5	source/router3.py File Reference	34
<b>Index</b>		<b>35</b>





# Chapter 1

## Namespace Index

### 1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<b>client</b>	9
<b>graphIO</b>	11
<b>hardy_cross</b>	11
<b>kpi_calculator</b>	13
<b>router3</b>	13



## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

graphIO.display_graph . . . . .	15
graphIO.graph_reader . . . . .	17
graphIO.graph_writer . . . . .	20
kpi_calculator.kpi_calculator . . . . .	23
object	
hardy_cross.HardyCross . . . . .	20
router3.Router . . . . .	26



## Chapter 3

# Class Index

### 3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<b>graphIO.display_graph</b> . . . . .	15
<b>graphIO.graph_reader</b> . . . . .	17
<b>graphIO.graph_writer</b> . . . . .	20
<b>hardy_cross.HardyCross</b> . . . . .	20
<b>kpi_calculator.kpi_calculator</b> . . . . .	23
<b>router3.Router</b> . . . . .	26



## Chapter 4

# File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

source/ <b>client.py</b> . . . . .	33
source/ <b>graphIO.py</b> . . . . .	33
source/ <b>hardy_cross.py</b> . . . . .	34
source/ <b>kpi_calculator.py</b> . . . . .	34
source/ <b>router3.py</b> . . . . .	34





## Chapter 5

# Namespace Documentation

### 5.1 client Namespace Reference

#### Functions

- def **render\_vtk** (file\_name)
- def **vesuvio\_example** ()
- def **paesi\_example** ()
- def **casdetude** ()
- def **casdetude\_dinardo** ()
- def **adjacency\_matrix** ()
- def **automatic\_partitioning** ()

#### Variables

- **PROJECT\_PATH** = os.path.dirname(os.path.abspath(\_\_file\_\_))
- int **last\_slash** = 0

#### 5.1.1 Function Documentation

##### 5.1.1.1 adjacency\_matrix()

```
def client.adjacency_matrix ( )
```

Imports a graph from an adjacency matrix and performs some spectra operations on it

#### 5.1.1.2 automatic\_partitioning()

```
def client.automatic_partitioning ( )
```

Import a graph from an adjacency and runs the luvain community partitionig algorithm to find communities.  
The result is show at screen

#### 5.1.1.3 casdetude()

```
def client.casdetude ( )
```

Imports the topograpy of the city of Monterusciello and automatically design the water supply network

#### 5.1.1.4 casdetude\_dinardo()

```
def client.casdetude_dinardo ( )
```

Imports the topograpy of the city of Monterusciello and automatically design the water supply network

#### 5.1.1.5 paesi\_example()

```
def client.paesi_example ( )
```

Example of water distribution network partitioning.  
The result is exported to a shapefile

#### 5.1.1.6 render\_vtk()

```
def client.render_vtk (
    file_name )
```

Renders a vtk file

#### 5.1.1.7 vesuvio\_example()

```
def client.vesuvio_example ( )
```

Example where the routing capabilities of the program are shown.  
A topography of the vesuvio area is imported from a vtk file and the shortest path no it is calculated.  
The result is than exported to vtk so that can be seen in paraview

### 5.1.2 Variable Documentation

#### 5.1.2.1 last\_slash

```
client.last_slash = 0
```

#### 5.1.2.2 PROJECT\_PATH

```
client.PROJECT_PATH = os.path.dirname(os.path.abspath(__file__))
```

## 5.2 graphIO Namespace Reference

### Classes

- class **display\_graph**
- class **graph\_reader**
- class **graph\_writer**

## 5.3 hardy\_cross Namespace Reference

### Classes

- class **HardyCross**

### Functions

- def **add\_string\_from\_list** (string\_list)
- def **flow\_correction\_dq** (df\_hf, df\_hf\_q)
- def **j\_loss\_10atm** (pipe\_diameter, flow\_rate)
- def **diameter\_from\_available** (theoretical\_diameter)
- def **diameter** (flow\_rate, velocity\_for\_diameter=0.8, show=0)
- def **velocity** (flow\_rate, pipe\_diameter)

### 5.3.1 Function Documentation

#### 5.3.1.1 `add_string_from_list()`

```
def hardy_cross.add_string_from_list (
    string_list )
```

```
:rtype : str
```

#### 5.3.1.2 `diameter()`

```
def hardy_cross.diameter (
    flow_rate,
    velocity_for_diameter = 0.8,
    show = 0 )
```

Q: flow rate [l/s]

velocity\_for\_diameter: flow velocity m/s

D: diameter [mm]

#### 5.3.1.3 `diameter_from_available()`

```
def hardy_cross.diameter_from_available (
    theoretical_diameter )
```

#### 5.3.1.4 `flow_correction_dq()`

```
def hardy_cross.flow_correction_dq (
    df_hf,
    df_hf_q )
```

```
:type df_hf_q: float
```

```
:type df_hf: float
```

#### 5.3.1.5 j\_loss\_10atm()

```
def hardy_cross.j_loss_10atm (
    pipe_diameter,
    flow_rate )
```

```
:param pipe_diameter: float
:param flow_rate: float
d: diameter [mm]
q: flow rate [l/s]
j: losses [m]
```

#### 5.3.1.6 velocity()

```
def hardy_cross.velocity (
    flow_rate,
    pipe_diameter )
```

## 5.4 kpi\_calculator Namespace Reference

### Classes

- class **kpi\_calculator**

## 5.5 router3 Namespace Reference

### Classes

- class **Router**



## Chapter 6

# Class Documentation

### 6.1 graphIO.display\_graph Class Reference

#### Public Member Functions

- def **\_\_init\_\_** (self, **graph**)
- def **distance2D** (self, nodei, nodej)
- def **distance3D** (self, nodei, nodej)
- def **coord2D** (self, G)
- def **display\_mesh** (self)
- def **display\_path** (self, path)

#### Public Attributes

- **graph**

#### 6.1.1 Detailed Description

This class implements some usefull display function to plot the acqueduct a graph

#### 6.1.2 Constructor & Destructor Documentation

##### 6.1.2.1 \_\_init\_\_()

```
def graphIO.display_graph.__init__ (
    self,
    graph )
```

## 6.1.3 Member Function Documentation

### 6.1.3.1 coord2D()

```
def graphIO.display_graph.coord2D (
    self,
    G )
```

Returns 2D coordinates of the nodes of self.graph

### 6.1.3.2 display\_mesh()

```
def graphIO.display_graph.display_mesh (
    self )
```

### 6.1.3.3 display\_path()

```
def graphIO.display_graph.display_path (
    self,
    path )
```

Display the mesh with a path marked on it

### 6.1.3.4 distance2D()

```
def graphIO.display_graph.distance2D (
    self,
    nodei,
    nodej )
```

Computes the cartesian norme in 2D



### 6.1.3.5 distance3D()

```
def graphIO.display_graph.distance3D (
    self,
    nodei,
    nodej )
```

Computes the cartesian norme in 3D

## 6.1.4 Member Data Documentation

### 6.1.4.1 graph

```
graphIO.display_graph.graph
```

The documentation for this class was generated from the following file:

- source/ **graphIO.py**

## 6.2 graphIO.graph\_reader Class Reference

### Public Member Functions

- def **\_\_init\_\_** (self, **graph**)
- def **avg** (self, node\_list)
- def **read\_shp\_building** (self, file\_name)
- def **row\_chuncker** (self, array, p\_array)
- def **read\_shp** (self, file\_name, point\_file=None)
- def **read\_adjacency** (self, filename)
- def **read\_epanet** (self, filename)

### Public Attributes

- **graph**

### 6.2.1 Detailed Description

This class provides the functions for graph reading from a multitude of formats

### 6.2.2 Constructor & Destructor Documentation

#### 6.2.2.1 `__init__()`

```
def graphIO.graph_reader.__init__ (
    self,
    graph )
```

### 6.2.3 Member Function Documentation

#### 6.2.3.1 `avg()`

```
def graphIO.graph_reader.avg (
    self,
    node_list )
```

#### 6.2.3.2 `read_adjacency()`

```
def graphIO.graph_reader.read_adjacency (
    self,
    filename )
```

#### 6.2.3.3 `read_epanet()`

```
def graphIO.graph_reader.read_epanet (
    self,
    filename )
```

#### 6.2.3.4 `read_shp()`

```
def graphIO.graph_reader.read_shp (
    self,
    file_name,
    point_file = None )
```

Generates a networkx.DiGraph from shapefiles. Point geometries are translated into nodes, lines into edges. Coordinate tuples are used as keys. Attributes are preserved, line geometries are simplified into start and end coordinates. Accepts a single shapefile or directory of many shapefiles.

"The Esri Shapefile or simply a shapefile is a popular geospatial vector data format for geographic information systems software."

#### Parameters

-----

path : file or string

File, directory, or filename to read.

simplify: bool

If ``True``, simplify line geometries to start and end coordinates.

If ``False``, and line feature geometry has multiple segments, the non-geometric attributes for that feature will be repeated for each edge comprising that feature.

#### Returns

-----

G : NetworkX graph

#### Examples

-----

```
>>> G=nx.read_shp('test.shp') # doctest: +SKIP
```

#### References

-----

.. [1] <http://en.wikipedia.org/wiki/Shapefile>

### 6.2.3.5 read\_shp\_bilding()

```
def graphIO.graph_reader.read_shp_bilding (
    self,
    file_name )
```

### 6.2.3.6 row\_chuncker()

```
def graphIO.graph_reader.row_chuncker (
    self,
    array,
    p_array )
```

## 6.2.4 Member Data Documentation

### 6.2.4.1 graph

graphIO.graph\_reader.graph

The documentation for this class was generated from the following file:

- source/ **graphIO.py**

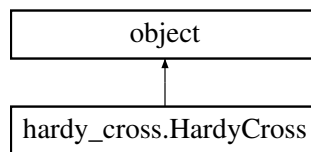
## 6.3 graphIO.graph\_writer Class Reference

The documentation for this class was generated from the following file:

- source/ **graphIO.py**

## 6.4 hardy\_cross.HardyCross Class Reference

Inheritance diagram for hardy\_cross.HardyCross:



### Public Member Functions

- def **\_\_init\_\_** (self, **loops**)
- def **compute\_pipe\_diameter\_of\_each\_loop** (self)
- def **compute\_velocities\_of\_each\_loop** (self)
- def **sort\_edge\_names** (self)
- def **locate\_common\_loops** (self)
- def **run\_hc** (self)
- def **save\_flows\_to\_file** (self)

### Public Attributes

- **runs**
- **threshold**
- **loops**
- **common\_loops**
- **delta\_Qs**
- **velocities**
- **new\_D**
- **smallest\_flow\_rate**
- **max\_velocity**
- **min\_velocity**

### 6.4.1 Constructor & Destructor Documentation

#### 6.4.1.1 \_\_init\_\_()

```
def hardy_cross.HardyCross.__init__ (
    self,
    loops )
```

## 6.4.2 Member Function Documentation

### 6.4.2.1 compute\_pipe\_diameter\_of\_each\_loop()

```
def hardy_cross.HardyCross.compute_pipe_diameter_of_each_loop (
    self )
```

### 6.4.2.2 compute\_velocities\_of\_each\_loop()

```
def hardy_cross.HardyCross.compute_velocities_of_each_loop (
    self )
```

### 6.4.2.3 locate\_common\_loops()

```
def hardy_cross.HardyCross.locate_common_loops (
    self )
```

this method locates the common edges and for each loop it creates a sparce matrix/array where the common edge/ is symbolized by the number 1. each matrix later will be multiplied (dot) by the delta\_Qs.  
:return: None

### 6.4.2.4 run\_hc()

```
def hardy_cross.HardyCross.run_hc (
    self )
```

### 6.4.2.5 save\_flows\_to\_file()

```
def hardy_cross.HardyCross.save_flows_to_file (
    self )
```

### 6.4.2.6 sort\_edge\_names()

```
def hardy_cross.HardyCross.sort_edge_names (
    self )
```

### 6.4.3 Member Data Documentation

#### 6.4.3.1 common\_loops

`hardy_cross.HardyCross.common_loops`

#### 6.4.3.2 delta\_Qs

`hardy_cross.HardyCross.delta_Qs`

#### 6.4.3.3 loops

`hardy_cross.HardyCross.loops`

#### 6.4.3.4 max\_velocity

`hardy_cross.HardyCross.max_velocity`

#### 6.4.3.5 min\_velocity

`hardy_cross.HardyCross.min_velocity`

#### 6.4.3.6 new\_D

`hardy_cross.HardyCross.new_D`

#### 6.4.3.7 runs

`hardy_cross.HardyCross.runs`

#### 6.4.3.8 smallest\_flow\_rate

`hardy_cross.HardyCross.smallest_flow_rate`

#### 6.4.3.9 threshold

`hardy_cross.HardyCross.threshold`

#### 6.4.3.10 velocities

`hardy_cross.HardyCross.velocities`

The documentation for this class was generated from the following file:

- source/ **hardy\_cross.py**

## 6.5 kpi\_calculator.kpi\_calculator Class Reference

### Public Member Functions

- def **\_\_init\_\_**(self, **wdn**)
- def **print\_kpi**(self)

### Public Attributes

- **available\_power**
- **dissipated\_power**
- **nodes\_power**
- **MIN**
- **MAX**

### Static Public Attributes

- **wdn** = nx.Graph()
- int **available\_power** = 0
- int **dissipated\_power** = 0
- list **MIN** = []
- list **MAX** = []
- list **MEAN** = []
- list **SQM** = []

### 6.5.1 Detailed Description

This class computes performance indicators for a solved water distribuion network. Various types of indicators are available:

- Energy indicators of the network
  - available\_power
  - dissipated\_power
- Statistical indicators of nodes head per cluster
  - min
  - max
  - mean
  - mean square error

### 6.5.2 Constructor & Destructor Documentation

#### 6.5.2.1 \_\_init\_\_()

```
def kpi_calculator.kpi_calculator.__init__ (
    self,
    wdn )
```

### 6.5.3 Member Function Documentation

#### 6.5.3.1 print\_kpi()

```
def kpi_calculator.kpi_calculator.print_kpi (
    self )
```

### 6.5.4 Member Data Documentation

#### 6.5.4.1 available\_power [1/2]

```
int kpi_calculator.kpi_calculator.available_power = 0 [static]
```

#### 6.5.4.2 available\_power [2/2]

```
kpi_calculator.kpi_calculator.available_power
```



**6.5.4.3** `dissipated_power` [1/2]

```
int kpi_calculator.kpi_calculator.dissipated_power = 0 [static]
```

**6.5.4.4** `dissipated_power` [2/2]

```
kpi_calculator.kpi_calculator.dissipated_power
```

**6.5.4.5** `MAX` [1/2]

```
list kpi_calculator.kpi_calculator.MAX = [] [static]
```

**6.5.4.6** `MAX` [2/2]

```
kpi_calculator.kpi_calculator.MAX
```

**6.5.4.7** `MEAN`

```
list kpi_calculator.kpi_calculator.MEAN = [] [static]
```

**6.5.4.8** `MIN` [1/2]

```
list kpi_calculator.kpi_calculator.MIN = [] [static]
```

**6.5.4.9** `MIN` [2/2]

```
kpi_calculator.kpi_calculator.MIN
```

**6.5.4.10** `nodes_power`

```
kpi_calculator.kpi_calculator.nodes_power
```

#### 6.5.4.11 SQM

```
list kpi_calculator.kpi_calculator.SQM = [] [static]
```

#### 6.5.4.12 wdn

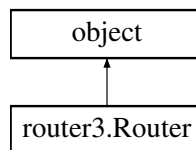
```
kpi_calculator.kpi_calculator.wdn = nx.Graph() [static]
```

The documentation for this class was generated from the following file:

- source/ **kpi\_calculator.py**

## 6.6 router3.Router Class Reference

Inheritance diagram for router3.Router:



### Public Member Functions

- def **\_\_init\_\_** (self, topo\_file=None, building\_file=None, adjacency\_metrix=None)
- def **write2shp** (self, G, filename)
- def **write2epanet** (self, G, filename)
- def **write2vtk** (self, G, filename)
- def **add\_node\_unique** (self, new\_node, new\_attributes)
- def **read\_vtk** (self, file\_name)
- def **distance** (self, nodei, nodej)
- def **shortest\_path** (self, node1, node2)
- def **path\_lenght** (self, path)
- def **is\_sourcesink** (self, node)
- def **compute\_source\_matrix** (self)
- def **design\_minimal\_aqueduct** (self, G, weight='dist')
- def **complete\_graph** (self, G)
- def **mesh\_graph** (self, G, weight)
- def **graphToEdgeMatrix** (self, G)
- def **cluster** (self, G)
- def **design\_aqueduct** (self, LEVEL=0)
- def **louvain\_clustering** (self, G, weight=None)
- def **routing** (self, n1, n2)
- def **solve** (self, G)

## Static Public Attributes

- string **CLASS\_NAME** = "Router"
- string **CLASS\_AUTHOR** = "Marcello Vaccarino"
- **graph** = nx.Graph()
- **sinksource\_graph** = nx.Graph()
- **acqueduct\_1level** = nx.Graph()
- **acqueduct\_2level** = nx.Graph()
- **acqueduct** = nx.Graph()

## 6.6.1 Constructor & Destructor Documentation

### 6.6.1.1 `__init__()`

```
def router3.Router.__init__ (
    self,
    topo_file = None,
    building_file = None,
    adjacency_metrix = None )
```

## 6.6.2 Member Function Documentation

### 6.6.2.1 `add_node_unique()`

```
def router3.Router.add_node_unique (
    self,
    new_node,
    new_attributes )
```

Grants that the node added is unique with respect to the pos attribute equality relationship.  
Deprecated since the node is a coordinates tuple

### 6.6.2.2 `cluster()`

```
def router3.Router.cluster (
    self,
    G )
```

Finds the clusters in a graph and returns a tuple with labels and nodes centers

#### 6.6.2.3 complete\_graph()

```
def router3.Router.complete_graph (
    self,
    G )
```

Completes in place the given graph

#### 6.6.2.4 compute\_source\_matrix()

```
def router3.Router.compute_source_matrix (
    self )
```

Yelds the subgraph having sink and sources as nodes  
an the shortest path between each one of them as edge

#### 6.6.2.5 design\_aqueduct()

```
def router3.Router.design_aqueduct (
    self,
    LEVEL = 0 )
```

Performs automatic water network distribution design.

#### 6.6.2.6 design\_minimal\_aqueduct()

```
def router3.Router.design_minimal_aqueduct (
    self,
    G,
    weight = 'dist' )
```

Computes the skeletonisation of the graph with minimum spanning tree algorithm

#### 6.6.2.7 distance()

```
def router3.Router.distance (
    self,
    nodei,
    nodej )
```

Computes euclidian distance between two nodes

#### 6.6.2.8 graphToEdgeMatrix()

```
def router3.Router.graphToEdgeMatrix (
    self,
    G )
```

Returns the adjacency matrix of the graph

#### 6.6.2.9 is\_sourcesink()

```
def router3.Router.is_sourcesink (
    self,
    node )
```

Given a node as in the `networkx.Graph.nodes(data=1)`  
returns 1 if the node is a sink or a source, 0 elsewhere

#### 6.6.2.10 louvain\_clustering()

```
def router3.Router.louvain_clustering (
    self,
    G,
    weight = None )
```

Performs Water Distribution Networks partitioning according to the method shown in the article from Qingzhou Zhang; Zheng Yi Wu; Ming Zhao; Jingyao Qi; Yuan Huang; and Hongbin Zhao in the article "Automatic Partitioning of Water Distribution Networks Using Multiscale Community Detection and Multiobjective

#### 6.6.2.11 mesh\_graph()

```
def router3.Router.mesh_graph (
    self,
    G,
    weight )
```

Given a graph, returns a graph with the same nodes connected according the gabriel definition of neighbourhood  
complexity is  $(\text{len}(G.\text{nodes}))^3$

#### 6.6.2.12 path\_lenght()

```
def router3.Router.path_lenght (
    self,
    path )
```

Given a path on the graph returns the lenght of the path in the  
unit the coordinats are expressed

#### 6.6.2.13 read\_vtk()

```
def router3.Router.read_vtk (
    self,
    file_name )
```

Import a graph from a vtk file

#### 6.6.2.14 routing()

```
def router3.Router.routing (
    self,
    n1,
    n2 )
```

#### 6.6.2.15 `shortest_path()`

```
def router3.Router.shortest_path (
    self,
    node1,
    node2 )
```

Calculates the shortest path on `self.graph`.  
Returns the path as a sequence of traversed nodes

#### 6.6.2.16 `solve()`

```
def router3.Router.solve (
    self,
    G )
```

Find the pipes diameter for a given aqueduct topology respecting the following constraints:

- pipes velocities between 0.5 and 1 m/s
- pipes diameters commercially available

#### 6.6.2.17 `write2epanet()`

```
def router3.Router.write2epanet (
    self,
    G,
    filename )
```

Exports an aqueduct to an inp epanet input file

#### 6.6.2.18 `write2shp()`

```
def router3.Router.write2shp (
    self,
    G,
    filename )
```

Exports a graph to a shapefile to be visualized with qgis

#### 6.6.2.19 write2vtk()

```
def router3.Router.write2vtk (
    self,
    G,
    filename )
```

Exports a graph to a vtk file to be visualized with paraview

### 6.6.3 Member Data Documentation

#### 6.6.3.1 acqueduct

```
router3.Router.acqueduct = nx.Graph() [static]
```

#### 6.6.3.2 acqueduct\_1level

```
router3.Router.acqueduct_1level = nx.Graph() [static]
```

#### 6.6.3.3 acqueduct\_2level

```
router3.Router.acqueduct_2level = nx.Graph() [static]
```

#### 6.6.3.4 CLASS\_AUTHOR

```
string router3.Router.CLASS_AUTHOR = "Marcello Vaccarino" [static]
```

#### 6.6.3.5 CLASS\_NAME

```
string router3.Router.CLASS_NAME = "Router" [static]
```

#### 6.6.3.6 graph

```
router3.Router.graph = nx.Graph() [static]
```

#### 6.6.3.7 sinksource\_graph

```
router3.Router.sinksource_graph = nx.Graph() [static]
```

The documentation for this class was generated from the following file:

- source/ **router3.py**



## Chapter 7

# File Documentation

### 7.1 source/client.py File Reference

#### Namespaces

- **client**

#### Functions

- **def client.render\_vtk** (file\_name)
- **def client.vesuvio\_example** ()
- **def client.paesi\_example** ()
- **def client.casdetude** ()
- **def client.casdetude\_dinardo** ()
- **def client.adjacency\_matrix** ()
- **def client.automatic\_partitioning** ()

#### Variables

- **client.PROJECT\_PATH** = os.path.dirname(os.path.abspath(\_\_file\_\_))
- **int client.last\_slash** = 0

### 7.2 source/graphIO.py File Reference

#### Classes

- **class graphIO.graph\_reader**
- **class graphIO.graph\_writer**
- **class graphIO.display\_graph**

#### Namespaces

- **graphIO**

## 7.3 source/hardy\_cross.py File Reference

### Classes

- class **hardy\_cross.HardyCross**

### Namespaces

- **hardy\_cross**

### Functions

- def **hardy\_cross.add\_string\_from\_list** (string\_list)
- def **hardy\_cross.flow\_correction\_dq** (df\_hf, df\_hf\_q)
- def **hardy\_cross.j\_loss\_10atm** (pipe\_diameter, flow\_rate)
- def **hardy\_cross.diameter\_from\_available** (theoretical\_diameter)
- def **hardy\_cross.diameter** (flow\_rate, velocity\_for\_diameter=0.8, show=0)
- def **hardy\_cross.velocity** (flow\_rate, pipe\_diameter)

## 7.4 source/kpi\_calculator.py File Reference

### Classes

- class **kpi\_calculator.kpi\_calculator**

### Namespaces

- **kpi\_calculator**

## 7.5 source/router3.py File Reference

### Classes

- class **router3.Router**

### Namespaces

- **router3**

# Index

- `__init__`
  - `graphIO::display_graph`, 15
  - `graphIO::graph_reader`, 17
  - `hardy_cross::HardyCross`, 20
  - `kpi_calculator::kpi_calculator`, 24
  - `router3::Router`, 27
- `acqueduct`
  - `router3::Router`, 32
- `acqueduct_1level`
  - `router3::Router`, 32
- `acqueduct_2level`
  - `router3::Router`, 32
- `add_node_unique`
  - `router3::Router`, 27
- `add_string_from_list`
  - `hardy_cross`, 12
- `adjacency_matrix`
  - `client`, 9
- `automatic_partitioning`
  - `client`, 9
- `available_power`
  - `kpi_calculator::kpi_calculator`, 24
- `avg`
  - `graphIO::graph_reader`, 18
- `CLASS_AUTHOR`
  - `router3::Router`, 32
- `CLASS_NAME`
  - `router3::Router`, 32
- `casdetude`
  - `client`, 10
- `casdetude_dinardo`
  - `client`, 10
- `client`, 9
  - `adjacency_matrix`, 9
  - `automatic_partitioning`, 9
  - `casdetude`, 10
  - `casdetude_dinardo`, 10
  - `last_slash`, 11
  - `PROJECT_PATH`, 11
  - `paesi_example`, 10
  - `render_vtk`, 10
  - `vesuvio_example`, 10
- `cluster`
  - `router3::Router`, 27
- `common_loops`
  - `hardy_cross::HardyCross`, 22
- `complete_graph`
  - `router3::Router`, 27
- `compute_pipe_diameter_of_each_loop`
  - `hardy_cross::HardyCross`, 21
- `compute_source_matrix`
  - `router3::Router`, 28
- `compute_velocities_of_each_loop`
  - `hardy_cross::HardyCross`, 21
- `coord2D`
  - `graphIO::display_graph`, 16
- `delta_Qs`
  - `hardy_cross::HardyCross`, 22
- `design_aqueduct`
  - `router3::Router`, 28
- `design_minimal_aqueduct`
  - `router3::Router`, 28
- `diameter`
  - `hardy_cross`, 12
- `diameter_from_available`
  - `hardy_cross`, 12
- `display_mesh`
  - `graphIO::display_graph`, 16
- `display_path`
  - `graphIO::display_graph`, 16
- `dissipated_power`
  - `kpi_calculator::kpi_calculator`, 24, 25
- `distance`
  - `router3::Router`, 28
- `distance2D`
  - `graphIO::display_graph`, 16
- `distance3D`
  - `graphIO::display_graph`, 16
- `flow_correction_dq`
  - `hardy_cross`, 12
- `graph`
  - `graphIO::display_graph`, 17
  - `graphIO::graph_reader`, 19
  - `router3::Router`, 32
- `graphIO.display_graph`, 15
- `graphIO.graph_reader`, 17
- `graphIO.graph_writer`, 20
- `graphIO::display_graph`
  - `__init__`, 15
  - `coord2D`, 16
  - `display_mesh`, 16
  - `display_path`, 16
  - `distance2D`, 16
  - `distance3D`, 16
  - `graph`, 17

- graphIO::graph\_reader
  - \_\_init\_\_, 17
  - avg, 18
  - graph, 19
  - read\_adjacency, 18
  - read\_epanet, 18
  - read\_shp, 18
  - read\_shp\_building, 19
  - row\_chuncker, 19
- graphIO, 11
- graphToEdgeMatrix
  - router3::Router, 29
- hardy\_cross, 11
  - add\_string\_from\_list, 12
  - diameter, 12
  - diameter\_from\_available, 12
  - flow\_correction\_dq, 12
  - j\_loss\_10atm, 12
  - velocity, 13
- hardy\_cross.HardyCross, 20
- hardy\_cross::HardyCross
  - \_\_init\_\_, 20
  - common\_loops, 22
  - compute\_pipe\_diameter\_of\_each\_loop, 21
  - compute\_velocities\_of\_each\_loop, 21
  - delta\_Qs, 22
  - locate\_common\_loops, 21
  - loops, 22
  - max\_velocity, 22
  - min\_velocity, 22
  - new\_D, 22
  - run\_hc, 21
  - runs, 22
  - save\_flows\_to\_file, 21
  - smallest\_flow\_rate, 22
  - sort\_edge\_names, 21
  - threshold, 23
  - velocities, 23
- is\_sourcesink
  - router3::Router, 29
- j\_loss\_10atm
  - hardy\_cross, 12
- kpi\_calculator, 13
- kpi\_calculator.kpi\_calculator, 23
- kpi\_calculator::kpi\_calculator
  - \_\_init\_\_, 24
  - available\_power, 24
  - dissipated\_power, 24, 25
  - MAX, 25
  - MEAN, 25
  - MIN, 25
  - nodes\_power, 25
  - print\_kpi, 24
  - SQM, 25
  - wdn, 26
- last\_slash
  - client, 11
- locate\_common\_loops
  - hardy\_cross::HardyCross, 21
- loops
  - hardy\_cross::HardyCross, 22
- louvain\_clustering
  - router3::Router, 29
- MAX
  - kpi\_calculator::kpi\_calculator, 25
- MEAN
  - kpi\_calculator::kpi\_calculator, 25
- MIN
  - kpi\_calculator::kpi\_calculator, 25
- max\_velocity
  - hardy\_cross::HardyCross, 22
- mesh\_graph
  - router3::Router, 29
- min\_velocity
  - hardy\_cross::HardyCross, 22
- new\_D
  - hardy\_cross::HardyCross, 22
- nodes\_power
  - kpi\_calculator::kpi\_calculator, 25
- PROJECT\_PATH
  - client, 11
- paesi\_example
  - client, 10
- path\_lenght
  - router3::Router, 30
- print\_kpi
  - kpi\_calculator::kpi\_calculator, 24
- read\_adjacency
  - graphIO::graph\_reader, 18
- read\_epanet
  - graphIO::graph\_reader, 18
- read\_shp
  - graphIO::graph\_reader, 18
- read\_shp\_building
  - graphIO::graph\_reader, 19
- read\_vtk
  - router3::Router, 30
- render\_vtk
  - client, 10
- router3, 13
- router3.Router, 26
- router3::Router
  - \_\_init\_\_, 27
  - acqueduct, 32
  - acqueduct\_1level, 32
  - acqueduct\_2level, 32
  - add\_node\_unique, 27
  - CLASS\_AUTHOR, 32
  - CLASS\_NAME, 32
  - cluster, 27

- complete\_graph, 27
- compute\_source\_matrix, 28
- design\_aqueduct, 28
- design\_minimal\_aqueduct, 28
- distance, 28
- graph, 32
- graphToEdgeMatrix, 29
- is\_sourcesink, 29
- louvain\_clustering, 29
- mesh\_graph, 29
- path\_lenght, 30
- read\_vtk, 30
- routing, 30
  - router3::Router, 30
- row\_chuncker
  - graphIO::graph\_reader, 19
- run\_hc
  - hardy\_cross::HardyCross, 21
- runs
  - hardy\_cross::HardyCross, 22
- SQM
  - kpi\_calculator::kpi\_calculator, 25
- save\_flows\_to\_file
  - hardy\_cross::HardyCross, 21
- shortest\_path
  - router3::Router, 30
- sinksources\_graph
  - router3::Router, 32
- smallest\_flow\_rate
  - hardy\_cross::HardyCross, 22
- solve
  - router3::Router, 31
- sort\_edge\_names
  - hardy\_cross::HardyCross, 21
- source/client.py, 33
- source/graphIO.py, 33
- source/hardy\_cross.py, 34
- source/kpi\_calculator.py, 34
- source/router3.py, 34
- threshold
  - hardy\_cross::HardyCross, 23
- velocities
  - hardy\_cross::HardyCross, 23
- velocity
  - hardy\_cross, 13
- vesuvio\_example
  - client, 10
- wdn
  - kpi\_calculator::kpi\_calculator, 26
  - write2epanet
    - router3::Router, 31
  - write2shp
    - router3::Router, 31
  - write2vtk
    - router3::Router, 31