Projet_P5C006

Generated by Doxygen 1.8.14

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

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 comunity partitionig algorithm to find communites.
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 topograpy 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_bilding** (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**

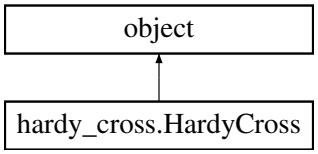
6.3 graphIO.graph_writer Class Reference

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.
Varius 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

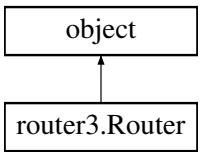
```
kpi_calculator.kpi_calculator.wdn = nx.Graph()  [static]
```

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 touple
```

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 (len(G.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 acqueduct 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