## Structure of modules

This description applies to the version committed to github on June 8th, 2020. Features added after this day may not be included in this document

Link:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Pyswmm\_GUI

This module contains all widgets of the Graphical User Interface (i.e. the physical and visible components of the GUI)

Run this script to run the tool from python.

The structure is the following:

#### \_\_init\_\_:

Creates the window and settings such as size, title, iconbitmap (the small icon in the top corner) etc.

Also executes all subsequent functions.

#### create\_widgets():

Here all visible components are defined.

First the top part is defined. This includes model selection and logo.

Then the notebook frame (the various tabs) are definded.

To add tabs it should be included here and afterwards content can be written.

The buttons in the bottom of the GUI is defined.

The content of each tab is created in the following part. This makes up most of the script.

#### if \_\_name\_\_ == "\_\_main\_\_":

In the very end of the document

Execute the program by starting the mainloop.

### Gui\_elements

This module contains all functions that are not visible in the GUI and not related to the actual computation of the simulations.

#### Run():

This is executed when the *Run* button is activated. Either the configuration file will be written, or the simulation will be run with the existing configuration file.

#### Write\_config():

Write all parameters that are given in the GUI to a configuration file with the same name as the model. This file is saved in *\NOAH\_RTC\_Tool\config\saved\_configs* and can be edited with any text editor. If *Overwrite existing configuration file* is not checked the simulation will run with the most recent version of this file and most input in the GUI is ignored.

If the *save config* button is pushed the configuration file is written without running the simulation.

#### Parameters():

This class contain all parameters that are used before the actual computation. This includes parameters that defines what kind of simulations that are run, that are shown in the GUI or changes the appearance of the GUI.

Also, parameters that are defined via radiobuttons or checkboxes needs to be defined here because these set an existing variable to a given value or string.

#### Tooltip/create\_Tooltip():

Creates a small text that appears when the cursor is above a widget. Useful when explanations are required.

#### Small functions:

A range of small functions that are used when changing states in the GUI based on input to the GUI.   
These are typically specific and only used on a few widgets each.

#### OpenFile():

Opens a dialog that allow the user to choose a SWMM model.

#### User\_msg():

The function is activated every time a simulation is run. Gives useful information about status of simulation, computation time etc.

#### Results\_plot():

Makes a plot with some data depending on the input.   
Also saves the plot as a .png in the output folder.

#### First\_step\_optimization\_plot():

Creates a plot of the first step of the optimization. Calls *Reults\_plot()*.

NB. The plots for the results are in GUI\_emelents.py while the text is in the *Optimizer* class in pyswmm\_Simulation.py.   
The functions related to the results are not very intuitively structured. Ideally a class is created with all functions related to the results so that these are streamlined and easier to edit. This is part of the future work.

### Pyswmm\_Simulation

This module contains all code that is related to the computation of SWMM models and processing of the results.

#### Optimizer:

This class contains all functions related to the optimization of the RTC.   
All following functions are methods within this class.

#### Init():

Initializes the class and parameters required before the optimization can begin. This includes creating an output directory for the results and choosing the correct optimizer.

#### read\_config():

This method reads the parameters from the most recent configuration file with the same name as the model.

The try/except ValueError: clause is only applied where the type is required to be float. This ensures that spaces can be left blank without causing an error. However, if the parameter is needed for the computation the error occurs at a later stage. This can be troublesome if it for instance occur after the simulation when the results are to be written since this will cause loss of results.

A way of validating the configuration file and checking that all required parameters are correct should be implemented in the future.

#### Redefine\_Timeseries():

This method changes the Timeseries in the SWMM model to the correct directory since it would else require that all external files are in the *lib* directory. Not sure whether this works correct!

#### Two\_step\_optimizer():

This is the actual optimizer. It contains two parts:

Part 1 is an initial screening of the parameter space defined by input in the Optimization tab in the GUI.   
This is used to save time in the optimization since the starting point is chosen more accurate.   
Also, this part allows the user to easily get an overview of whether RTC shows a potential or not.

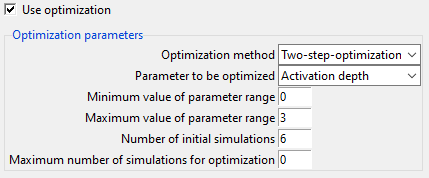
Part 2 is the “finetuning” of the optimization setting. It begins at the lowest point of the screening and uses a build in python optimizer to find the lowest point. This typically requires more simulations and computation time.

Part 2 will only be computed if the “Maximum number of simulations for the optimization” is above 0.

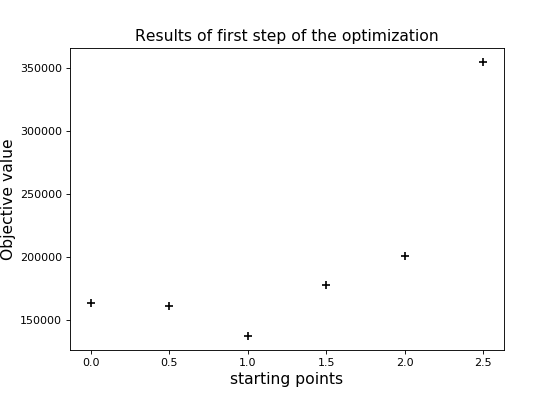
A final simulation is run to ensure that the saved .rpt and .out files are the optimal ones. This is not necessarily optimal if computation time is too long.

*Example:*

The node with the sensor is expected have a depth no higher than 3 meters and 6 simulations are used for the initial screening of the simulation. The GUI is shown below:



This results in the graph below where the objective value is computed for 6 equal intervals between 0 and 3.



If “Maximum number of simulations for the optimization” is greater than 0, the build-in optimizer would begin at 1.0 and terminate either when the lowest point is reached or when the given amount of simulations is computed.

#### optimized\_simulation():

This method computes each simulation. The input parameter x is the one that is optimized in the Two-step-optimizer and that is to be determined. If optimization is not applied the activation depth from the GUI is used.

The simulations differ depending on the actuator type. Pump is not tested properly yet.

#### Opjective functions:

The result processing that returns the objective value is computed in these functions.

This is either number of CSO events or volume.

#### write\_optimal\_result():

This writes a text file that is shown after the optimization. As mentioned, the structure of the results are not intuitive and should be improved.

## Adding parameters to the code

When new parameters are to be added to the python code the following steps must be done.

* Add the visible widgets where the parameters must be typed in in pyswmm\_GUI.py. (E.g. entry, radiobutton, Checkbutton etc.)
* Add the parameter in the *write\_config()* function in GUI\_elements.py. This ensures that the configuration file contains the parameter.
* Add the parameter in the *read\_config()* function in the *Optimizer* class in pyswmm\_Simulation.py. This ensures that the parameter can be used in the simulation.
* Add the parameter to the *parameters* class in GUI\_elements.py. ONLY NECESSARY IF the parameter is used before the actual computation. I.e. if it defines what kind of simulation that should be run, if it is shown somewhere in the GUI or if it changes the appearance of the GUI. (Examples are model name, model directory, whether optimization is applied.)
* Add the parameter in the actual computation in pyswmm\_Simulation.py.

## Calibration

### Requirements:

It is required to have a running SWMM model with the pipes connected in the flow direction (i.e. the water flows from the inlet node to the outlet node).