# AUTOMATED PRELIMINARY DESIGN OF URBAN DRAINAGE SYSTEMS – PROPOSAL

MAX LANGE – 5169402

## INDEX



## USED ABBREVIATIONS:

- UDS: Urban Drainage System
- OSM: OpenStreetMap
- SWMM: Storm Water Management Model

# 1. PROBLEM STATEMENT

When looking at a certain area of a map, it can be difficult to assess certain attributes about a possible urban drainage system (UDS) for that area without seeing the network in front of you (for example total size, dimensions of pipes, flowrates or total cost). Creating a preliminary design by hand can take a significant amount of time, especially if the area is large. It is possible to contact the relevant government agencies to access the existing UDS plans, but this could take time, or involve a cumbersome bureaucratic process. On top of this, the plans that you receive could be in a format which does not help in assessing the wanted attributes. These are all problems that many have addressed (Palumbo et al., 2014; Maurer et al., 2013; Eric W. Strecker & Wayne C. Huber, 2012; Wang & Wang, 2018), with all of them either devising their own tool or concluding that having a tool or quick algorithm to circumvent these obstacle will be beneficial.

# 2. OBJECTIVE

The Objective of this Bachelor End Project is to create a software tool in Python, which will be able to create a realistic preliminary design of a UDS, based on the existing road network of an area selected by the user. This tool will function as both a way of speeding up the initial assessment of a possible UDS for a certain area, and also as an initial stepping stone for a more detailed UDS design.

## 2.1 REQUIREMENTS

The software tool should be able to:

- Create a network of pipes, junction (manholes) and gullies for a given area based on the existing road network, and calculate the catchment area of each gully
- Determine the direction of flow in the pipes based on known outflow points provided by the user
- Calculate the diameter of the pipes based on significant rainfall parameters provided by the user
- Convert the designed network into the Storm Water Management Model (SWMM) (Rossman, 2022) format

The final report for this Bachelor End Project will closely follow the steps that the software takes (described in chapter 3) to come to its final design. This will then be followed by an evaluation of its accuracy, as well as an discussion into further possible addition or other uses for the program.

## 2.2 DEVELOPMENT SETUP

Python has been selected as the main programming language for the program. As a UDS can be visualized as a graph, graph theory and computation will be central to this project, in which python excels for both, and has high end capabilities. It is also the language with which I, as the creator of the software, have the most coding experience, and thus will prevent large time expenses because of the need to learn a new language.

A GitHub repository will be setup to keep track of the codebase, and to ease the development pipeline and issue tracking process. This repository will also contain an explanation (README) on how to use the software, and should be able to be used as a standalone software tool, without needing the final BEP report as reference.

## 2.3 EVALUATION CRITERIA

As the aim of the tool is to replicate the hand-made initial design of a UDS as realistically as possible, it may be difficult to properly asses it's success, as it depends on our definition of what a "realistic" UDS is. Naturally, attributes like the layout following the roads, diameters of the pipes not being disproportionately large or small, or the direction of flow being the wrong way around can easily be ascertained, but to see if the generated design actually matches a "realistic" UDS will take some extra steps.

For now the manner in which the software will be assessed for accuracy will be visual. By testing the software on areas for which (some of) the attributes that are calculated are known, the results can be compared with the real version. A better defined definition for the evaluation criteria is preferred, and will be looked in to later on in the project.

A more detailed outline of the major steps that the program will take to create an UDS are given below. Also laid out are possible additions to the software, should the initial version be completed ahead of schedule.

# 3. LAYOUT OF DESIGN STEPS

1. Extract road network data of a user selected area from OpenStreetMap

   User inputs:

   a. Bounding box of the area to run the program on (most northern, southern, eastern and western points)

   b. Maximum allowed distance of gullies from one another.

2. Create junctions, pipes and intermediate gullies network from road network data

3. Calculate catchment area for each gully

4. Let user set attributes of the network

   User inputs:

   a. Create (multiple) outflow points at handpicked nodes

   b. Set the design storm to which the initial diameter estimation will be done

   c. OPTIONAL: Set the maximum/minimum allowable diameter of the pipes

   d. Some form of defining the costs of the pipes used

5. Determine the direction of flow for the network

6. Calculate the initial pipe diameter based on the user-given parameters, and determine the total costs of the network based on this.

7. Implement the network into SWMM format (txt file)

## 3.1 STEP BY STEP EXPLANATION

### 1. EXTRACT ROAD NETWORK DATA FROM OPENSTREETMAP

When the user has input the coordinates of the area they want to inspect, OpenStreetMap (OSM) will be used to gather the road network data for that area. As OSM is a worldwide map, which is constantly being updated and corrected, it will provide a sufficiently accurate road network for almost any selected area. Python already has a module to easily download this data, called OSMNX (Boeing, 2017). This module is also able to present the node and line data in such a way that it can be convert to a useable format with relative easy, thus making it an logical choice.

## 2. CREATE JUNCTIONS, GULLIES AND PIPES NETWORK FROM ROAD NETWORK DATA

For a single section of road, it will be inspected to see if it requires (an) intermediate gullies(/gully). If so, the road section will be split into equal length pieces, and a gully will be inserted into each split. It will be assumed that each node (aka junction or end of a pipe) also have a gully alongside the usual manhole. This will create a network of pipes which will connect from gully to gully.

## 3. CALCULATE CATCHMENT AREA FOR EACH GULLY

To calculate the area of the subcatchment that each gully will be responsible for, the freud.locality module (Dice et al., 2019) will be used. This module provides the ability to construct a Voronoi diagram (Voronoi, 1908) of the network, and is also able to calculate the area of these regions, thus giving the total area closest to each gully.

## 4. LET USER SET OTHER ATTRIBUTES OF THE NETWORK

For the program to calculate the direction of flow, and give an initial calculation for the appropriate diameter and height of the pipes, more attributes of the network need to be known. The user will have the ability to select a single or multiple nodes to be outlet point(s) for the network. This will be enough to determine the direction of flow. To ascertain the other attributes, the user will have to input the L/s/ha of the storm for which they want the initial design to be. If desired, a maximum and/or minimum allowable diameter can also be specified (may be wanted for SWMM flooding simulations).

## 5. DETERMINE THE DIRECTION OF FLOW IN THE PIPES

To determine the direction of flow, Dijkstra's algorithm (Dijkstra, 1959) will be run for each point, for each outlet. Then the shortest path for each will be selected, which will determine the direction of flow. When taking elevation difference between point into consideration (See Possible Additions point 1), the weights of the paths or distance calculations of the algorithm need to be adjusted accordingly. How this will be done will be determined later on in the project, if it is possible to add the height data.

## 6. CALCULATE THE INITIAL DIAMETER OF THE PIPES, AND THE TOTAL COSTS

Assuming speed of water through the pipes is one meter per second during the significant rainfall, and the pipes are fully filled at this flow rate, the initial diameter of the pipes can easily be estimated by dividing the inflow by the speed of the water. The inflow can also be determined by summing up all the inflow from other pipes into the current pipe. With the diameter of the pipes determined, the initial costs assessment of the system can also be calculated with the cost value(s) provided by the user.

### 7. IMPLEMENT THE NETWORK INTO THE SWMM FORMAT

All the attributes that have been calculated in the steps listed above are values that the SWMM program can use / needs to have to run its simulations. SWMM uses a simple .txt file to store its data, and so a brand new SWMM data file can be created by using pythons own writing functions.

## 3.2 POSSIBLE ADDITIONS

All of the items that are listed below are possible additions to the software to either make it more realistic, add more attributes to the eventual SWMM format, or to enhance the user experience.

1. Extract height data from the internet to create a gravity based system. The network can take the actual elevation of the system to its advantage, as the path the water takes to get to the nearest outlet point can change depending on how much kinetic energy it losses or gains along the way. This fact can be used to simplify the drainage system, or determine other/better location for outlet points. If this can be achieved, it may also be possible to determine the optimal height of the pipes. If extracting height data from the internet turns out to be too cumbersome, it may also be decided to let the user manually set a slope steepness and orientation, to partially emulate the real world situation.

2. A User Interface. With the current design of the program, two user input moments are present during the operation of the software. These input moments, along with graphs of the results, could be displayed in multiple windows, instead of via the terminal.

# 4. CODE STRUCTURE

The step by step process of the program follows a linear fashion, but the user does have the ability to return to previous steps should they want to change certain inputs. Because of this reason, a Mediator design pattern will be implemented for the main structure of the program. This mediator, or main, will internally function according to a Chain of Responsibility structure, but will be able to jump back a certain amount of steps if told to do so by a user input. The module interactions and package dependencies can be seen in figure 1. A list of the main functionality of each module is given below the figure.

Although the design pattern of the software is fixed, the amount of modules and their individual functionalities may be subjected to change. If for example a module is too small to justify giving it a separate python file, it may be decided to merge it with another module with a similar functionality, and the reverse could be said for modules that are too large.

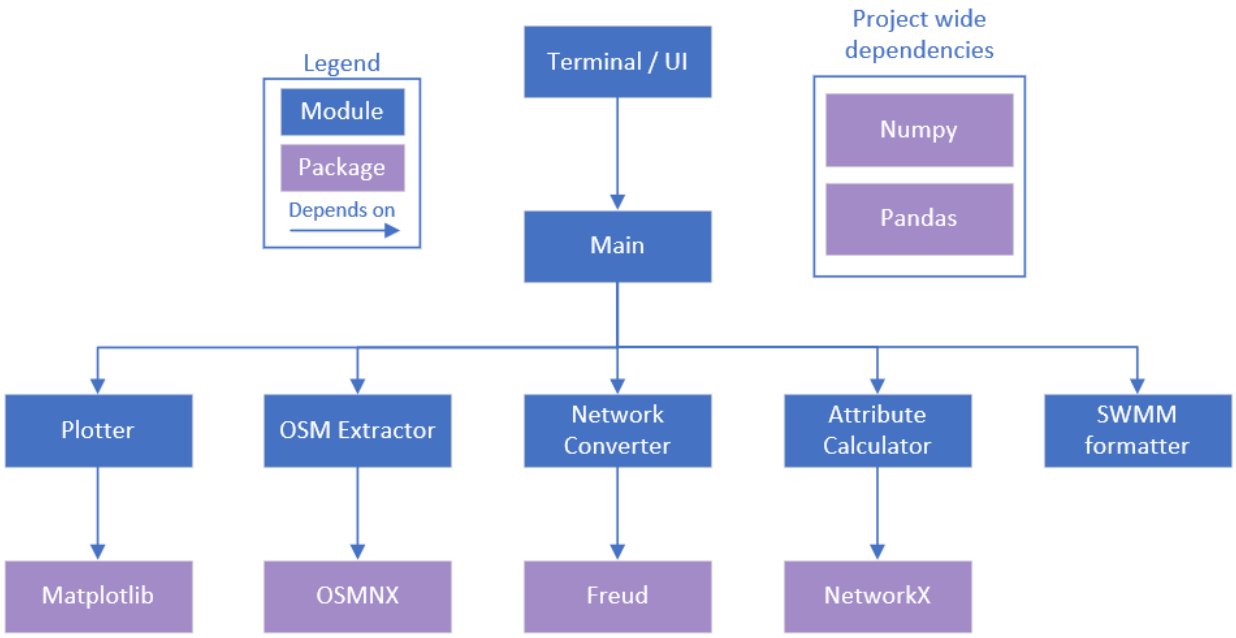


**Figure 1: Module structure and package dependencies**

- Terminal / UI: Displays the intermediate and end results of the designing process to the user, and also facilitate the input space.

- Main: Connect up all the modules, and facilitate the interactions between these modules and the terminal.

- Plotter: Contain the plotting functions which display the intermediate and end results of the program to the user. Enable other modules to purely consist of data manipulation or calculation functions.

- OSM Extractor: Extract the road network for the user specified area from OSM, and convert the data into a format which can be used by the network converter module.

- Network converter: Convert the OSM road network data into a network of pipes, junctions and gullies. Also calculate the catchment area for each gully using the Voronoi package of Freud.

- Attribute Calculator: Calculate the final specifications of the system, such as the direction of flow using Dijkstra's algorithm, and the initial diameters of the pipes and total costs.

- SWMM formatter: Convert the completed design into the SWMM .txt format.

## 5. SCHEDULE

Below are the planned schedules for the duration of the BEP period. They have been split into a Development schedule, and a surrounding requirements schedule to maintain readability.

**Development Schedule**

| Week Nr. | From-Till Dates | Tasks |
|---|---|---|
| 4.1 | 18-04 -> 22-04 | - Setup development plan<br>- Setup GitHub coding environment |
| 4.2 | 25-04 -> 29-04 | - Setup user input options for bounding box and gully limit<br>- Extract and reformat road data |
| 4.3 | 02-05 -> 06-05 | - Create network from road data<br>- Calculate Catchment area for each gully |
| 4.4 | 09-05 -> 13-05 | (Remains empty to focus on ethics and midway report) |
| 4.5 | 16-05 -> 20-05 | - Setup second user input options<br>- Determine direction of flow in pipes with Dijkstra's algorithm |
| 4.6 | 23-05 -> 27-05 | - Calculate initial diameter of pipes in network<br>- Convert to SWMM format |
| 4.7 | 30-05 -> 03-06 | - Wrap up code and final testing and bug fixing |
| 4.8 | 06-06 -> 10-06 | (Remains empty to focus on final report and presentation) |

**Surrounding requirements schedule**

| Week Nr. | From-Till Dates | Tasks | Deadline |
|---|---|---|---|
| 4.1 | 18-04 -> 22-04 | - Complete Information Literacy test<br>- Complete Proposal/Development plan | 25-03<br>25-03 |
| 4.2 | 25-04 -> 29-04 | - Work on / Complete ethics assignment | |
| 4.3 | 02-05 -> 06-05 | - Work on midway report | |
| 4.4 | 09-05 -> 13-05 | - Complete midway report<br>- Complete midway presentation | 13-05<br>13-05 |
| 4.5 | 16-05 -> 20-05 | - Complete peer review midway report and presentation<br>- Work on final report | 18/19-05 |
| 4.6 | 23-05 -> 27-05 | - Work on final report | |
| 4.7 | 30-05 -> 03-06 | - Complete final report<br>- Complete final presentation | 07-06<br>07-06 |
| 4.8 | 06-06 -> 10-06 | - Complete peer review final report and presentation<br>- Complete self-evaluation | T.B.D.<br>09/10-06 |

# BIBLIOGRAPHY


Boeing, G. (2017). OSMnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. *Computers, Environment and Urban Systems*, *65*, 126–139. https://doi.org/10.1016/j.compenvurbsys.2017.05.004

Dice, B., Ramasubramani, V., Harper, E., Spellings, M., Anderson, J., & Glotzer, S. (2019). Analyzing Particle Systems for Machine Learning and Data Visualization with freud. *Proceedings of the 18th Python in Science Conference*, 27–33. https://doi.org/10.25080/majora-7ddc1dd1-004

Dijkstra, E. W. (1959). A Note on Two Problems in Connexion with Graphs. *Numerische Mathematlk l*, 269–271. http://www-m3.ma.tum.de/foswiki/pub/MN0506/WebHome/dijkstra.pdf

Eric W. Strecker, & Wayne C. Huber. (2012, April 26). Global Solutions for Urban Drainage. *Global Solutions for Urban Drainage*. https://ascelibrary.org/doi/abs/10.1061/9780784406441

Maurer, M., Scheidegger, A., & Herlyn, A. (2013). Quantifying costs and lengths of urban drainage systems with a simple static sewer infrastructure model. *Urban Water Journal*, *10*(4), 268–280. https://doi.org/10.1080/1573062X.2012.731072

Palumbo, A., Cimorelli, L., Covelli, C., Cozzolino, L., Mucherino, C., & Pianese, D. (2014). Optimal design of urban drainage networks. *Civil Engineering and Environmental Systems*, *31*(1), 79–96. https://doi.org/10.1080/10286608.2013.820277

Rossman, L. (2022). Storm Water Management Model User's Manual Version. In *United States Enviromental Protection Agency*. United States Enviromental Protection Agency. https://www.epa.gov/water-research/storm-water-management-model-swmm

Voronoi, G. (1908). Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Premier mémoire. Sur quelques propriétés des formes quadratiques positives parfaites. *Journal Für Die Reine Und Angewandte Mathematik (Crelles Journal)*, *1908*(133), 97–102. https://doi.org/10.1515/crll.1908.133.97

Wang, S., & Wang, H. (2018). Extending the Rational Method for assessing and developing sustainable urban drainage systems. *Water Research*, *144*, 112–125. https://doi.org/10.1016/j.watres.2018.07.022