

# <sup>1</sup> Pysewer: A Python Library for Sewer Network Generation in Data Scarce Regions

<sup>3</sup> **Moritz Sanne<sup>1,2</sup>, Ganbaatar Khurelbaatar  <sup>1</sup>, Daneish Despot  <sup>1¶</sup>, Manfred van Afferden<sup>1</sup>, and Jan Friesen **

<sup>5</sup> 1 Centre for Environmental Biotechnology, Helmholtz Centre for Environmental Research GmbH – UFZ,  
<sup>6</sup> Permoserstraße 15 | 04318 Leipzig, Germany 2 Europace AG, Berlin, Germany ¶ Corresponding author

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

## Software

- [Review !\[\]\(b1b781be830eb908d845c527ab08d5f8\_img.jpg\)](#)
- [Repository !\[\]\(2176a4ba510fa27404d783166e891577\_img.jpg\)](#)
- [Archive !\[\]\(a3b1c8d49688274496e55f2751cb8993\_img.jpg\)](#)

---

Editor: [Open Journals !\[\]\(003082e50e3009141f59bd5df831749f\_img.jpg\)](#)

Reviewers:

- [@openjournals](#)

Submitted: 01 January 1970

Published: unpublished

## License

Authors of papers retain copyright<sup>18</sup> and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#))<sup>19</sup>.

## <sup>7</sup> Summary

<sup>8</sup> Pysewer is a network generator for sewer networks originally designed for rural settlements  
<sup>9</sup> in emerging countries with little or no wastewater infrastructure. The network generation  
<sup>10</sup> prioritises gravity flow in order to avoid pumping – which can be a source of failure and high  
<sup>11</sup> maintenance – where possible. The network dimensioning is based on dry-weather flow.

<sup>12</sup> Based on a few data sources, pysewer generates a complete network based on roads, building  
<sup>13</sup> locations, and elevation data. Global water consumption and population assumptions are  
<sup>14</sup> included to dimension the sewer diameters. Results are fully-connected sewer networks that  
<sup>15</sup> connect all buildings to one or several predefined wastewater treatment plant (WWTP) locations.  
<sup>16</sup> By default, the lowest point in the elevation data is set as the WWTP location. The resulting  
<sup>17</sup> network contains sewer diameters, building connections, as well as lifting or pumping stations  
<sup>18</sup> with pressurised pipes where necessary.

## Statement of need

<sup>20</sup> The sustainable management of water and sanitation has been defined as one of the UN's  
<sup>21</sup> sustainable development goals: SDG 6 ([UN-Water, 2018](#)). As of 2019, SDG 6 might not be  
<sup>22</sup> reached in 2030 despite the progress made, which means that more than half of the population  
<sup>23</sup> still lacks safely managed sanitation ([UN-Water, 2018](#)).

<sup>24</sup> In order to identify optimal wastewater management at the settlement level, it is necessary to  
<sup>25</sup> compare different central or decentral solutions. To achieve this, a baseline is required against  
<sup>26</sup> which other scenarios can be compared ([Khurelbaatar et al., 2021](#); [van Afferden et al., 2015](#)).  
<sup>27</sup> To this end, we developed pysewer – a tool that generates settlement-wide sewer networks,  
<sup>28</sup> which connect all the buildings within the settlement boundary or the region of interest to one  
<sup>29</sup> or more wastewater treatment plant locations.

<sup>30</sup> The core principle behind pysewer's development is based on numerical optimization methods.  
<sup>31</sup> These methods have been used for sewer network design since the 1960s ([Duque et al., 2020](#);  
<sup>32</sup> [Holland, 1966](#); [Li & Matthew, 1990](#); [Maurer et al., 2013](#); [Steele et al., 2016](#)), yet most require  
<sup>33</sup> detailed or inaccessible input data. Additionally, several Python-based tools employ graph  
<sup>34</sup> theory to optimize water distribution, water reuse, and wastewater master planning ([Calle et](#)  
<sup>35</sup> [al., 2023](#); [Friesen et al., 2023](#); [Momeni et al., 2023](#)). However, to our knowledge, there is  
<sup>36</sup> currently no well-documented and publicly available (open-source) Python package specifically  
<sup>37</sup> designed for generating sewer network layouts using graph theory. This gap is what pysewer  
<sup>38</sup> aims to fill.

<sup>39</sup> Pysewer is designed for data-scarce environments, utilizing only minimal data and global  
<sup>40</sup> assumptions – thus enabling transferability to a wide range of different regions. At the same

time, *a priori* data sources can be substituted with high-resolution data and site-specific information such as local water consumption and population data to enhance its accuracy and utility in specific contexts. The generated networks can then be exported (i.e., as a geopackage (.gpkg) or shapefile (.shp)) in order to utilise the results in preliminary planning stages, initial cost estimations, scenario development processes or for further comparison to decentral solutions where the network can be modified. The option to include several treatment locations also enables users to already plan decentralised networks or favour treatment locations (i.e., due to local demands or restrictions).

## 49 Functionality and key features

50 Pysewer's concept is built upon network science, where we combine algorithmic optimisation  
 51 using graph theory with sewer network engineering design to generate a sewer network layout.  
 52 In the desired layout, all buildings are connected to a wastewater treatment plant (WWTP)  
 53 through a sewer network, which utilises the terrain to prioritise gravity flow in order to minimise  
 54 the use of pressure sewers. Addressing the intricate challenge of generating sewer network  
 55 layouts, particularly in data-scarce environments, is at the forefront of our objectives. Our  
 56 approach, therefore, leans heavily towards utilising data that can be easily acquired for a  
 57 specific area of interest. Thus, we deploy the following data as input to autonomously generate  
 58 a sewer network, with a distinct prioritisation towards gravity flow.

- 59 1. Digital Elevation Model (DEM) – to derive the elevation profile and understand topo-  
 60 graphic details such as the lowest point (sinks) within the area of interest.
- 61 2. Existing road network data – Preferred vector data format in the form of LineString to  
 62 map and utilise current infrastructure pathways.
- 63 3. Building locations – defined by x, y coordinate points, these points represent service  
 64 requirement locations and identify the connection to the network.
- 65 4. Site-specific water consumption and population data – to plan/size hydraulic elements  
 66 of the sewer network and estimate the sewage flow.

67 The core functionalities of pysewer include transforming the minimal inputs into an initial  
 68 network graph—the foundation for the ensuing design and optimisation process; the generation  
 69 of a gravity flow-prioritised sewer network—identifying the most efficient network paths and  
 70 positions of the pump and lift stations where required; and the visualisation and exporting  
 71 of the generated network—allowing visual inspection of the sewer network attributes and  
 72 export of the generated sewer network. [Figure 1](#) provides a visual guide of the distinct yet  
 73 interconnected modules within pysewer.

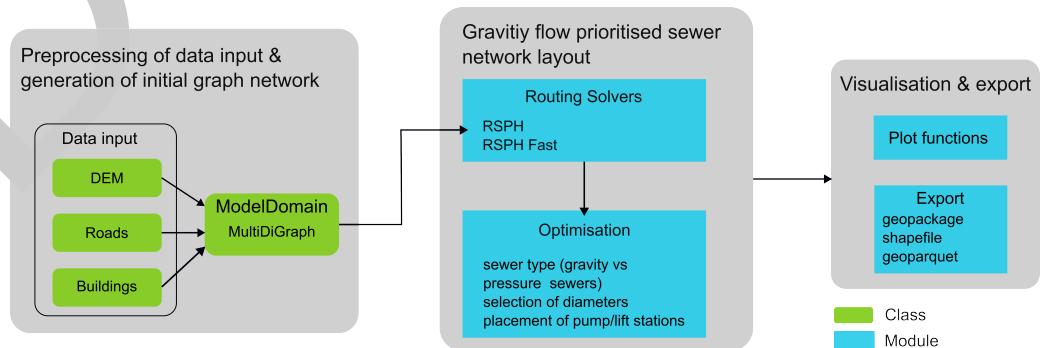


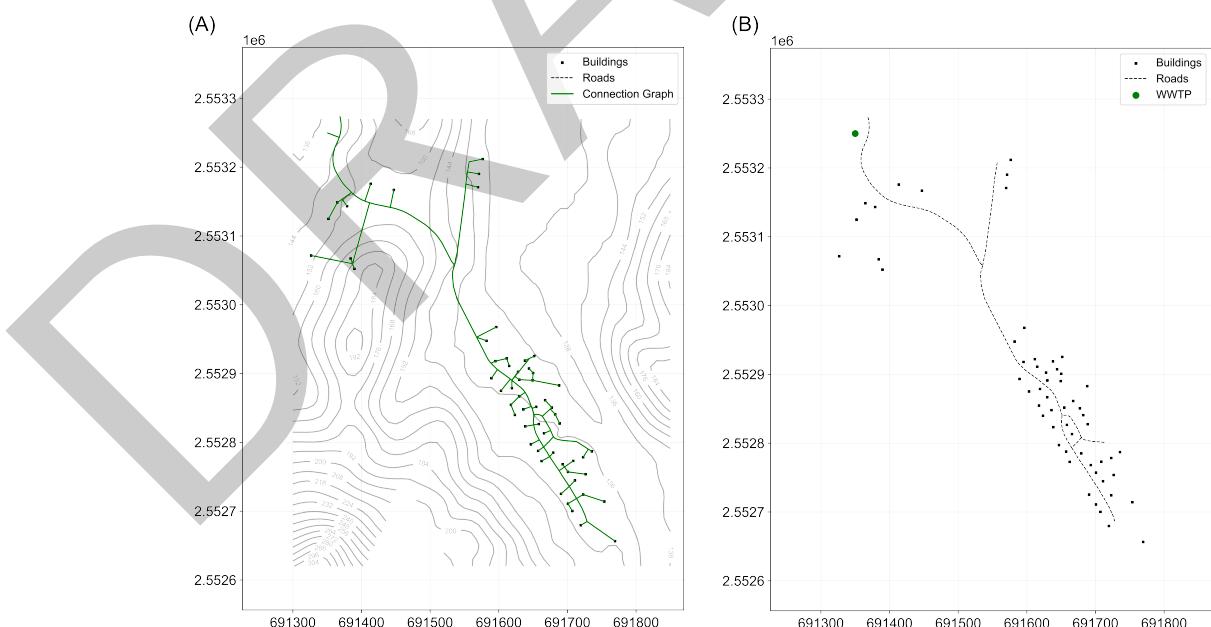
Figure 1: Pysewer's modular workflow

## 74 Preprocessing and initial network generation

75 In the preprocessing module, the roads, buildings, and the DEM must all be projected into the  
 76 same coordinate reference system (CRS). The road and building data input must be in the  
 77 form of either a geopandas ([Jordahl et al., 2020](#)) GeoDataFrame or a str which specifies the  
 78 path to a file with vector formats such shapefile (.shp), geojson (.geojson) or geopackage  
 79 (.gpkg). As for the DEM, the preferred format is a geotiff (.tif). Roads, Buildings and DEM  
 80 classes are used to transform the raw data formats into the required format (i.e., geopandas  
 81 GeoDataFrame) to create the initial graph network (NetworkX, ([Hagberg et al., 2008](#))), where  
 82 nodes represent crucial points such as junctions or buildings and edges to simulate potential  
 83 sewer lines. The following measures ensure that the initial layout aligns with the road network  
 84 and that there is serviceability to all buildings within the area of interest:

- 85 ▪ “Connecting” buildings to the street network using the connect buildings method. This  
 86 method adds nodes to the graph to connect the buildings in the network using the  
 87 building points.
- 88 ▪ Creation of “virtual roads”. Buildings which are not directly connected to the road  
 89 network are connected by finding the closest edge to the building, which is then marked  
 90 as the closest edge. The nodes are then disconnected from the edges and are added to  
 91 the initial connection graph network.
- 92 ▪ Simplifying the street network for more efficient graph traversal.
- 93 ▪ Setting of the collection point or Wastewater Treatment Plant (WWTP). By default,  
 94 the lowest elevation point in the region of interest is set as the location(s) of the WWTP.  
 95 Users can manually define the location of the WWTP by using the add\_sink method.

96 After preprocessing, all relevant data is stored as a MultiDiGraph to allow for asymmetric edge  
 97 values (e.g., elevation profile and subsequently costs). [Figure 2](#) demonstrates the required  
 98 data, its preprocessing and the generation of the initial graph network.



**Figure 2:** Pysewer preprocessing. Topographic map with the connection graph resulting from the instantiation of the ModelDomain class (A). Sewer network layout requirements: existing building, roads, and collection point (WWTP) (B).

## 99 Generating a gravity flow-prioritise sewer network

100 Within the computational framework of pysewer, the routing and optimisation modules function  
 101 as the principal mechanisms for synthesising the sewer network. The objective of the routing  
 102 module is to identify the paths through the network, starting from the sink. The algorithm  
 103 approximates the directed Steiner tree (the Steiner arborescence) (Hwang & Richards, 1992)  
 104 between all sources and the sink by using a repeated shortest path heuristic (RSPH). The routing  
 105 module has two solvers to find estimates for the underlying minimum Steiner arborescence  
 106 tree problem; these are:

- 107 1. The RSPH solver iteratively connects the nearest unconnected node (regarding distance  
   108 and pump penalty) to the closest connected network node. The solver can account for  
   109 multiple sinks and is well-suited to generate decentralised network scenarios.
- 110 2. The RSPH Fast solver derives the network by combining all shortest paths to a single  
   111 sink. It is faster but only allows for a single sink.

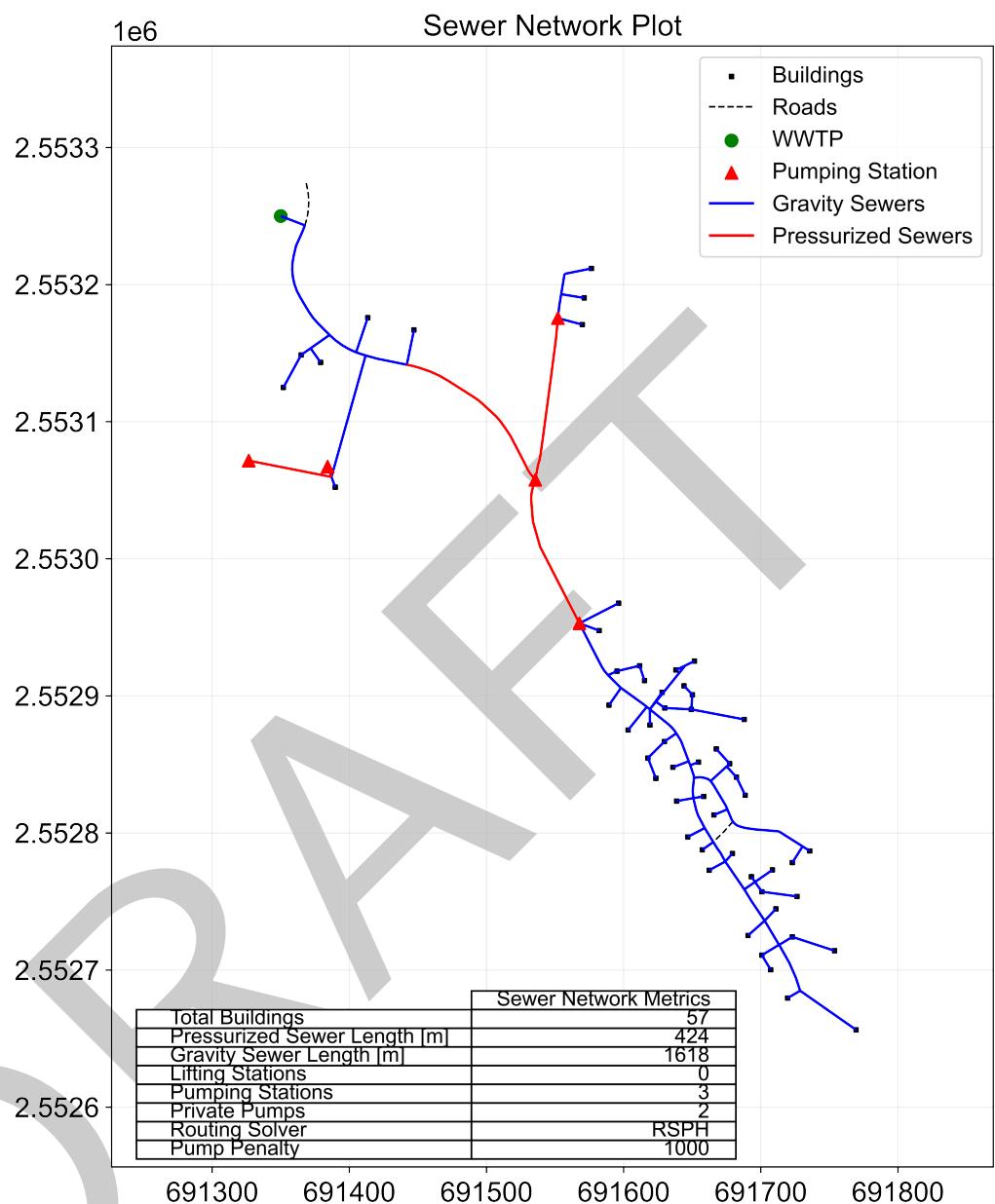
112 In a nutshell, these solvers work by navigating through the connection graph (created using  
 113 the generate\_connection\_graph method of the preprocessing module). This method first  
 114 simplifies the connection graph by removing any self-loops and setting trench depth node  
 115 attributes to 0. It then calculates key parameters such as geometry, distance, profile, initial  
 116 edge weights (needed for placing pump stations), and elevation attributes for each edge and  
 117 node. The shortest path between the subgraph and terminal nodes in the connection graph is  
 118 found using Dijkstra's Shortest Path Algorithm (Dijkstra, 1959). The RSPH solver repeatedly  
 119 finds the shortest path between the subgraph nodes and the closest terminal node, adding the  
 120 path to the sewer graph and updating the subgraph nodes and terminal nodes. Terminal nodes  
 121 refer to the nodes in the connection graph that need to be connected to the sink. On the other  
 122 hand, subgraph nodes are the nodes in the directed routed Steiner tree. These are initially  
 123 set to the sink nodes and are updated as the RSPH solver is applied to find the shortest path  
 124 between the subgraph and the terminal nodes. This way, all terminal nodes are eventually  
 125 connected to the sink.

126 Subsequently, the optimisation module takes the preliminary network generated by the routing  
 127 module and refines it by assessing and incorporating the hydraulic elements of the sewer  
 128 network. Here, the hydraulic parameters of the sewer network are calculated. The calculation  
 129 focuses on the placement of pump or lifting stations on linear sections between road junctions.  
 130 It considers the following three cases:

- 131 1. Terrain does not allow for gravity flow to the downstream node (this check uses the  
   132 needs\_pump attribute from the preprocessing to reduce computational load)—placement  
   133 of a pump station is required.
- 134 2. Terrain does not require a pump, but the lowest inflow trench depth is too low for  
   135 gravitational flow—placement of a lift station is required.
- 136 3. Gravity flow is possible within given constraints—the minimum slope is achieved, no  
   137 pump or lifting station is required.

138 As our tool strongly focuses on prioritising gravity flow, a high pump penalty is applied to  
 139 minimise the length of the pressure sewers. The pumping penalty expressed as the edge weight  
 140 is relative to the trench depth required to achieve minimum slope to achieve self-cleaning  
 141 velocities in a gravity sewer. The maximum trench depth  $t_{\max}$  required to achieve the minimum  
 142 slope is set at  $t_{\max} = 8m$  in the default settings of pysewer. When there is a need to dig  
 143 deeper than this predefined value, then a pump is required.

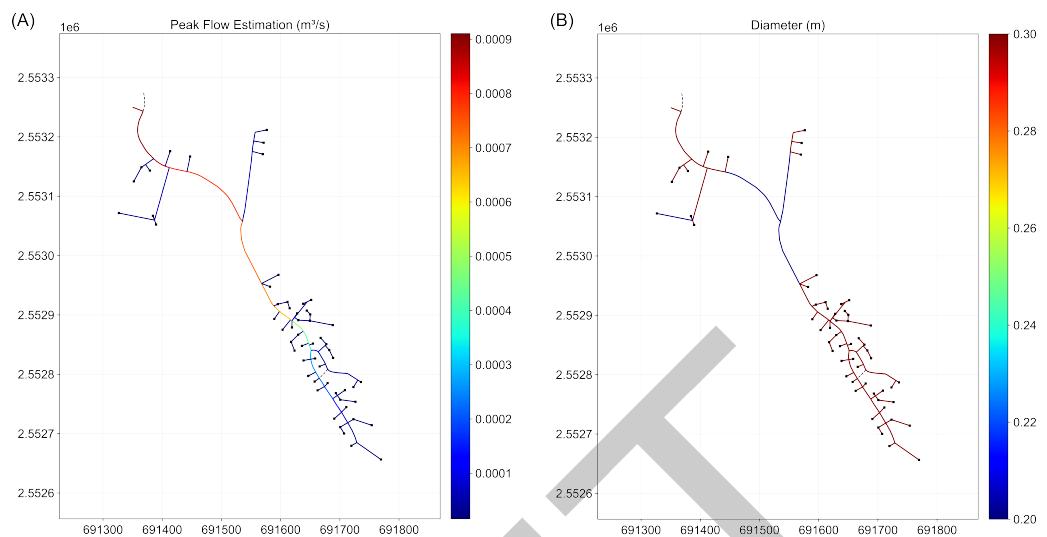
144 The optimisation module also facilitates the selection of the diameters to be used in the  
 145 network and peak flow estimation, as well as the key sewer attributes such as the number of  
 146 pump or lifting stations, the length of pressure and gravity sewers, which can be visualised  
 147 and exported for further analysis. Figure 3 shows an example of a final sewer network layout  
 148 generated after running the calculation of the hydraulics parameters.



**Figure 3:** Pysewer optimisation. Final layout of the sewer network.

#### 149 Visualising and exporting the generated sewer network

150 The plotting and exporting module generates visual and geodata outputs. It renders the  
 151 optimised network design onto a visual map, offering users an intuitive insight into the  
 152 proposed infrastructure. Sewer network attributes such as the estimated peak flow, the  
 153 selected pipe diameter (exemplified in [Figure 4](#)) and the trench profile are provided in the  
 154 final GeoDataFrame. They can be exported as a geopackage (.gpkg) or shapefile (.shp) file,  
 155 facilitating further analysis and detailed reporting in other geospatial platforms.



**Figure 4:** Pysewer visualisation. Attributes of the sewer network layout. Peak flow estimation (A), Pipe diameters selected (B)

## 156 Acknowledgement

157 M.S. and J.F. were supported by the MULTISOURCE project, which received funding from  
158 the European Union's Horizon 2020 program under grant agreement 101003527. G.K. and  
159 D.D. were supported by the WATERUN project, which was funded from the European Union's  
160 Horizon 2020 program under grant agreement 101060922. We thank Ronny Gey from the  
161 UFZ Research Data Management (RDM) group for reviewing the Git repository.

## 162 Software citations

163 Pysewer was written in Python 3.10.6 and used a suite of open-source software packages that  
164 aided the development process:

- 165     ■ Geopandas 0.8.1 ([Jordahl et al., 2020](#))
- 166     ■ NetworkX 3.1 ([Hagberg et al., 2008](#))
- 167     ■ Rasterio 1.2.10 ([Gillies & others, 2021](#))
- 168     ■ Numpy 1.25.2 ([Harris et al., 2020](#))
- 169     ■ Matplotlib 3.7.1 ([Hunter, 2007](#))
- 170     ■ Scikit-learn 1.0.2 ([Pedregosa et al., 2011](#))
- 171     ■ GDAL 3.0.2 ([GDAL/OGR contributors, 2019](#))

## 172 Author contributions

173 Conceptualisation: J.F., G.K., and M.v.A.; methodology: J.F., M.S., and D.D.; software  
174 development: M.S. and D.D.; writing – original draft: D.D.; writing – review & editing: D.D.,  
175 J.F., M.S., G.K., and M.v.A.

## 176 References

- 177 Calle, E., Martínez, D., Buttiglieri, G., Corominas, L., Farreras, M., Saló-Grau, J., Vilà, P.,  
178 Pueyo-Ros, J., & Comas, J. (2023). Optimal design of water reuse networks in cities

- 179 through decision support tool development and testing. *Npj Clean Water*, 6(1), Article 1.  
180 <https://doi.org/10.1038/s41545-023-00222-4>
- 181 Dijkstra, E. W. (1959). A Note on Two Problems in Connexion with Graphs. *Numerische  
182 Mathematik*, 1(1), 269–271. <https://doi.org/10.1007/BF01386390>
- 183 Duque, N., Duque, D., Aguilar, A., & Saldarriaga, J. (2020). Sewer Network Layout Selection  
184 and Hydraulic Design Using a Mathematical Optimization Framework. *Water*, 12(12),  
185 Article 12. <https://doi.org/10.3390/w12123337>
- 186 Friesen, J., Sanne, M., Khurelbaatar, G., & Afferden, M. van. (2023). “OCTOPUS” principle  
187 reduces wastewater management costs through network optimization and clustering. *One  
188 Earth*, 6(9), 1227–1234. <https://doi.org/10.1016/j.oneear.2023.08.005>
- 189 GDAL/OGR contributors. (2019). *GDAL/OGR Geospatial Data Abstraction software Library*  
190 (Version v3.0.2) [Computer software]. Open Source Geospatial Foundation. <https://doi.org/10.5281/zenodo.5884351>
- 191 Gillies, S., & others. (2021). *Rasterio: Geospatial raster I/O for Python programmers* (Version  
192 v1.2.10). Mapbox. <https://github.com/rasterio/rasterio>
- 193 Hagberg, A. A., Schult, D. A., & Swart, P. J. (2008). Exploring Network Structure, Dynamics,  
194 and Function using NetworkX. *Scipy*. <https://doi.org/10.25080/TCWV9851>
- 195 Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D.,  
196 Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk,  
197 M. H. van, Brett, M., Haldane, A., Río, J. F. del, Wiebe, M., Peterson, P., ... Oliphant,  
198 T. E. (2020). Array programming with NumPy. *Nature*, 585(7825), 357–362. <https://doi.org/10.1038/s41586-020-2649-2>
- 199 Holland, M. E. (1966). *Computer Models of Waste-Water Collection Systems* [PhD thesis].  
200 Harvard University.
- 201 Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science &  
202 Engineering*, 9(3), 90–95. <https://doi.org/10.1109/MCSE.2007.55>
- 203 Hwang, F. K., & Richards, D. S. (1992). Steiner Tree Problems. *Networks*, 22(1), 55–89.  
204 <https://doi.org/10.1002/net.3230220105>
- 205 Jordahl, K., Bossche, J. V. den, Fleischmann, M., Wasserman, J., McBride, J., Gerard, J.,  
206 Tratner, J., Perry, M., Badaracco, A. G., Farmer, C., Hjelle, G. A., Snow, A. D., Cochran,  
207 M., Gillies, S., Culbertson, L., Bartos, M., Eubank, N., maxalbert, Bilogur, A., ... Leblanc,  
208 F. (2020). *Geopandas* (Version v0.8.1). Zenodo. <https://doi.org/10.5281/zenodo.3946761>
- 209 Khurelbaatar, G., Al Marzuqi, B., Van Afferden, M., Müller, R. A., & Friesen, J. (2021).  
210 Data Reduced Method for Cost Comparison of Wastewater Management Scenarios – Case  
211 Study for Two Settlements in Jordan and Oman. *Frontiers in Environmental Science*, 9.  
212 <https://doi.org/10.3389/fenvs.2021.626634>
- 213 Li, G., & Matthew, R. G. S. (1990). New Approach for Optimization of Urban Drainage  
214 Systems. *Journal of Environmental Engineering*, 116(5), 927–944. [https://doi.org/10.1061/\(ASCE\)0733-9372\(1990\)116:5\(927\)](https://doi.org/10.1061/(ASCE)0733-9372(1990)116:5(927))
- 215 Maurer, M., Scheidegger, A., & Herlyn, A. (2013). Quantifying costs and lengths of urban  
216 drainage systems with a simple static sewer infrastructure model. *Urban Water Journal*,  
217 10(4), 268–280. <https://doi.org/10.1080/1573062X.2012.731072>
- 218 Momeni, A., Chauhan, V., Bin Mahmoud, A., Piratla, K. R., & Safro, I. (2023). Generation  
219 of Synthetic Water Distribution Data Using a Multiscale Generator-Optimizer. *Journal  
220 of Pipeline Systems Engineering and Practice*, 14(1). <https://doi.org/10.1061/jpsea2.pseng-1358>
- 221 Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel,  
222

- 226 M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau,  
227 D., Brucher, M., Perrot, M., & Duchesnay, Édouard. (2011). Scikit-learn: Machine  
228 Learning in Python. *Journal of Machine Learning Research*, 12(85), 2825–2830. <http://jmlr.org/papers/v12/pedregosa11a.html>
- 230 Steele, J. C., Mahoney, K., Karovic, O., & Mays, L. W. (2016). Heuristic Optimization Model  
231 for the Optimal Layout and Pipe Design of Sewer Systems. *Water Resources Management*,  
232 30(5), 1605–1620. <https://doi.org/10.1007/s11269-015-1191-8>
- 233 UN-Water. (2018). *Sustainable Development Goal 6: Synthesis Report 2018 on Water and  
234 Sanitation* (United Nations Publications). United Nations. ISBN: 978-92-1-101370-2
- 235 van Afferden, M., Cardona, J. A., Lee, M.-Y., Subah, A., & Müller, R. A. (2015). A New  
236 Approach to Implementing Decentralized Wastewater Treatment Concepts. *Water Science  
237 and Technology*, 72(11), 1923–1930. <https://doi.org/10.2166/wst.2015.393>

DRAFT