

¹ Pysewer: A Python Library for Sewer Network Generation in Data Scarce Regions

³ **Moritz Sanne^{1,2}, Ganbaatar Khurelbaatar  ¹, Daneish Despot  ^{1¶}, Manfred van Afferden¹, and Jan Friesen **

⁵ 1 Centre for Environmental Biotechnology, Helmholtz Centre for Environmental Research GmbH – UFZ,
⁶ 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Published: unpublished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 copyright¹⁸ and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#))¹⁹.

⁷ Summary

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

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

Statement of need

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

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

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

³⁹ Pysewer is designed for data-scarce environments, utilizing only minimal data and global
⁴⁰ 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 or shapefile) 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

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

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

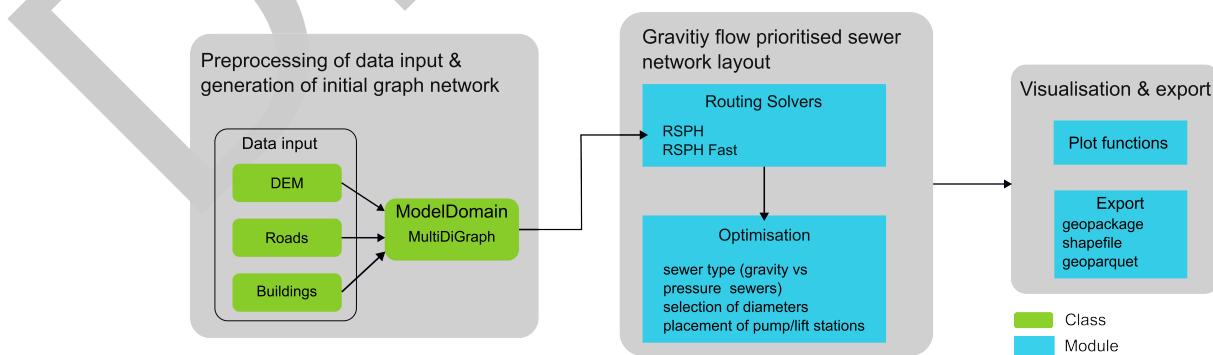


Figure 1: Pysewer's modular workflow

74 Preprocessing and initial network generation

In the preprocessing module, the roads, buildings and the DEM must all be projected in the same projection (CRS) and must be in the form of a geopandas ([Jordahl et al., 2020](#)) data

77 frame or a shapefile. Roads, Buildings and DEM classes are used to transform the raw data
 78 formats into the required format (i.e., geopandas data frame) to create the initial graph network
 79 (networkx, (Hagberg et al., 2008)), where nodes represent crucial points such as junctions or
 80 buildings and edges to simulate potential sewer lines. The following measures ensure that the
 81 initial layout aligns with the road network and that there is serviceability to all buildings within
 82 the area of interest:

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

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

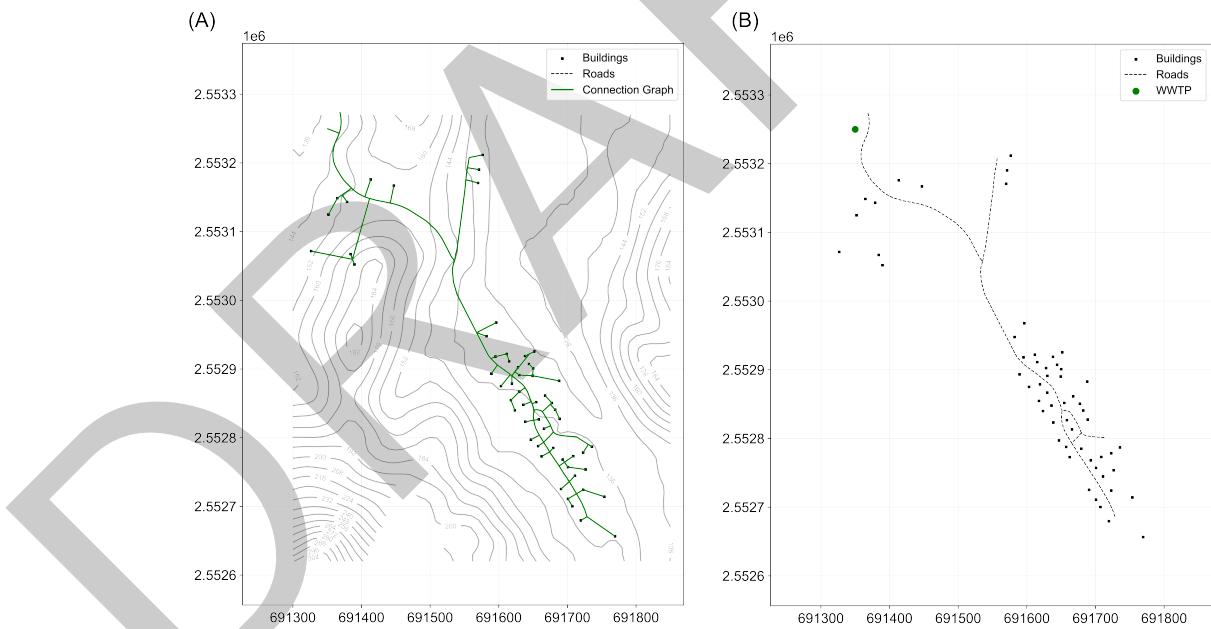


Figure 2: Pysewer preprocessing. Topographic map with the connection graph resulting from the instantiation of the ModelDomain class (A). Sewer network layout requirements: existing building, roads, and collection point (WWTP) (B).

97 Generating a gravity flow-prioritise sewer network

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

- 105 1. The RSPH solver iteratively connects the nearest unconnected node (regarding distance
106 and pump penalty) to the closest connected network node. The solver can account for
107 multiple sinks and is well-suited to generate decentralised network scenarios.
 - 108 2. The RSPH Fast solver derives the network by combining all shortest paths to a single
109 sink. It is faster but only allows for a single sink.
- 110 In a nutshell, these solvers work by navigating through the connection graph (created using the
111 generate_connection_graph method of the preprocessing module). This method simplifies
112 the connection graph, removes any self-loops, sets trench depth node attributes to 0, and
113 calculates the geometry, distance, profile, whether a pump is needed weight, and elevation
114 attributes for each edge and node. The shortest path between the subgraph and terminal
115 nodes in the connection graph is found using Dijkstra's Shortest Path Algorithm ([Dijkstra, 1959](#)). The RSPH solver repeatedly finds the shortest path between the subgraph nodes and
116 the closest terminal node, adding the path to the sewer graph and updating the subgraph
117 nodes and terminal nodes. Terminal nodes refer to the nodes in the connection graph that
118 need to be connected to the sink. On the other hand, subgraph nodes are the nodes in the
119 directed routed Steiner tree. These are initially set to the sink nodes and are updated as the
120 RSPH solver is applied to find the shortest path between the subgraph and the terminal nodes.
121 This way, all terminal nodes are eventually connected to the sink.
- 122 Subsequently, the optimisation module takes the preliminary network generated by the routing
123 module and refines it by assessing and incorporating the hydraulic elements of the sewer
124 network. Here, the hydraulic parameters of the sewer network are calculated. The calculation
125 focuses on the placement of pump or lifting stations on linear sections between road junctions.
126 It considers the following three cases:
- 127 1. Terrain does not allow for gravity flow to the downstream node (this check uses the
128 needs_pump attribute from the preprocessing to reduce computational load)—placement
129 of a pump station is required.
 - 130 2. Terrain does not require a pump, but the lowest inflow trench depth is too low for
131 gravitational flow—placement of a lift station is required.
- 132 Gravity flow is possible within given constraints—the minimum slope is achieved, no pump
133 or lifting station is required. As our tool strongly focuses on prioritising gravity flow, a high
134 pump penalty is applied to minimise the length of the pressure sewers. The pumping penalty
135 expressed as the edge weight is relative to the trench depth required to achieve minimum slope
136 to achieve self-cleaning velocities in a gravity sewer. The maximum trench depth t_{\max} required
137 to achieve the minimum slope is set at $t_{\max} = 8$ in the default settings of pysewer. When
138 there is a need to dig deeper than this predefined value, then a pump is required.
- 139 The optimisation module also facilitates the selection of the diameters to be used in the
140 network and peak flow estimation, as well as the key sewer attributes such as the number of
141 pump or lifting stations, the length of pressure and gravity sewers, which can be visualised
142 and exported for further analysis. [Figure 3](#) shows an example of a final sewer network layout
143 generated after running the calculation of the hydraulics parameters.

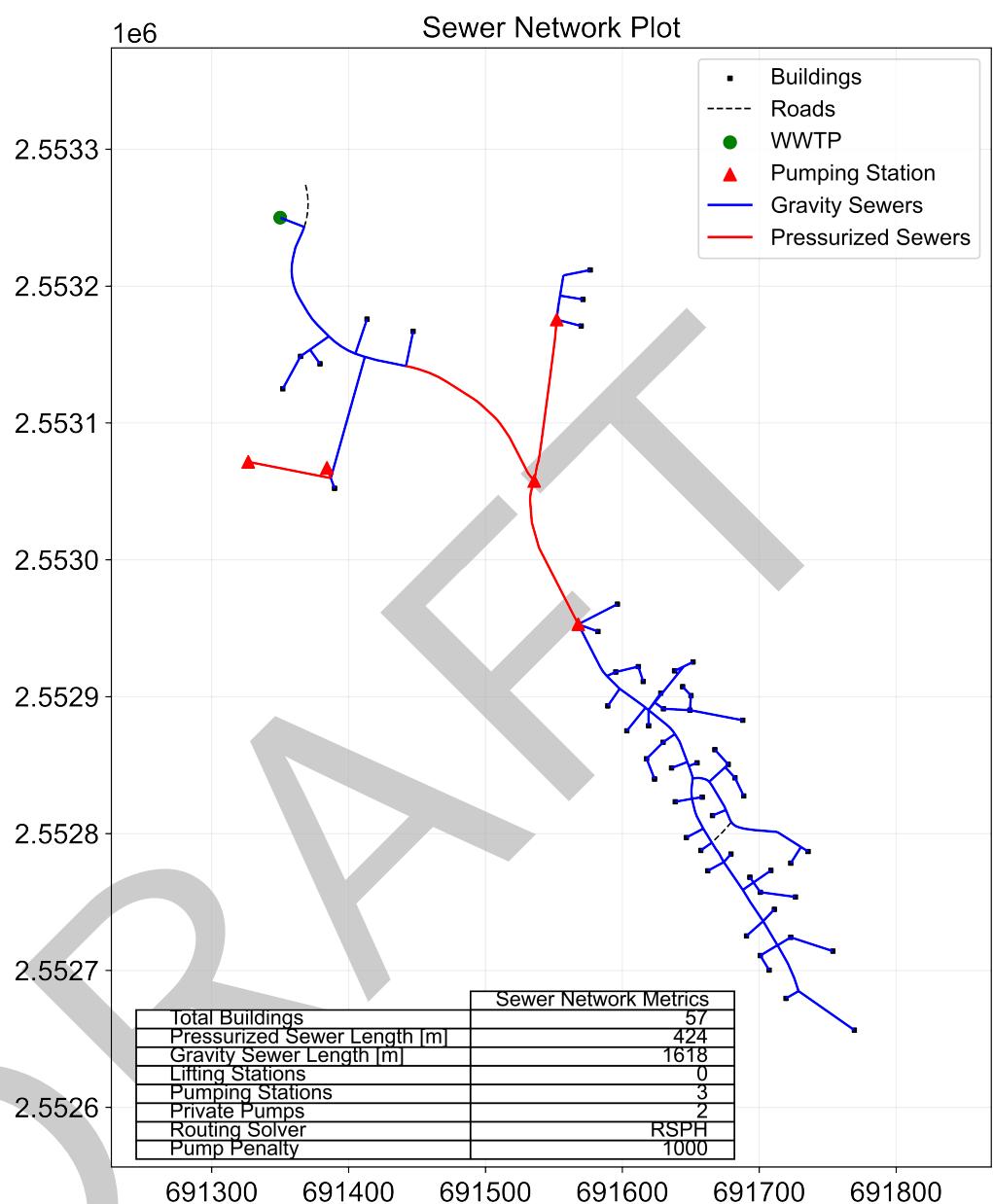


Figure 3: Pysewer optimisation. Final layout of the sewer network.

145 Visualising and exporting the generated sewer network

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

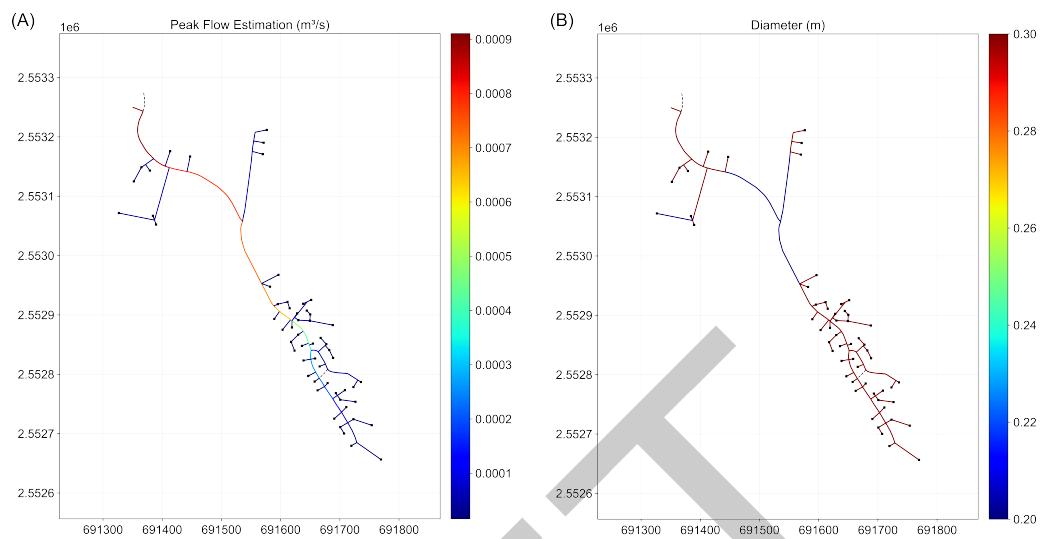


Figure 4: Pysewer visualisation. Attributes of the sewer network layout. Peak flow estimation (A), Pipe diameters selected (B)

152 Acknowledgement

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

158 Software citations

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

- 161 ■ Geopandas 0.9.0 ([Jordahl et al., 2020](#))
- 162 ■ Networkx 3.1 ([Hagberg et al., 2008](#))
- 163 ■ Numpy 1.25.2 ([Harris et al., 2020](#))
- 164 ■ Matplotlib 3.7.1 ([Hunter, 2007](#))
- 165 ■ Sklearn 1.0.2 ([Pedregosa et al., 2011](#))
- 166 ■ GDAL 3.0.2 ([GDAL/OGR contributors, 2023](#))

167 Author contributions

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

171 References

- 172 Calle, E., Martínez, D., Buttiglieri, G., Corominas, L., Farreras, M., Saló-Grau, J., Vilà, P.,
 173 Pueyo-Ros, J., & Comas, J. (2023). Optimal design of water reuse networks in cities
 174 through decision support tool development and testing. *Npj Clean Water*, 6(1), Article 1.
<https://doi.org/10.1038/s41545-023-00222-4>

- 176 Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische
177 Mathematik*, 1(1), 269–271. <https://doi.org/10.1007/BF01386390>
- 178 Duque, N., Duque, D., Aguilar, A., & Saldarriaga, J. (2020). Sewer network layout selection
179 and hydraulic design using a mathematical optimization framework. *Water*, 12(12), Article
180 12. <https://doi.org/10.3390/w12123337>
- 181 Friesen, J., Sanne, M., Khurelbaatar, G., & Afferden, M. van. (2023). “OCTOPUS” principle
182 reduces wastewater management costs through network optimization and clustering. *One
183 Earth*, 6(9), 1227–1234. <https://doi.org/10.1016/j.oneear.2023.08.005>
- 184 GDAL/OGR contributors. (2023). *GDAL/OGR geospatial data abstraction software library*.
185 Open Source Geospatial Foundation. <https://doi.org/10.5281/zenodo.5884351>
- 186 Hagberg, A. A., Schult, D. A., & Swart, P. J. (2008). Exploring network structure, dynamics,
187 and function using NetworkX. In G. Varoquaux, T. Vaught, & J. Millman (Eds.), *Proceedings
188 of the 7th python in science conference* (pp. 11–15).
- 189 Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D.,
190 Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk,
191 M. H. van, Brett, M., Haldane, A., Río, J. F. del, Wiebe, M., Peterson, P., ... Oliphant,
192 T. E. (2020). Array programming with NumPy. *Nature*, 585(7825), 357–362. <https://doi.org/10.1038/s41586-020-2649-2>
- 194 Holland, M. E. (1966). *Computer models of waste-water collection systems* [PhD thesis].
195 Harvard University.
- 196 Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science &
197 Engineering*, 9(3), 90–95. <https://doi.org/10.1109/MCSE.2007.55>
- 198 Hwang, F. K., & Richards, D. S. (1992). Steiner tree problems. *Networks*, 22(1), 55–89.
199 <https://doi.org/10.1002/net.3230220105>
- 200 Jordahl, K., Bossche, J. V. den, Fleischmann, M., Wasserman, J., McBride, J., Gerard,
201 J., Tratner, J., Perry, M., Badaracco, A. G., Farmer, C., Hjelle, G. A., Snow, A. D.,
202 Cochran, M., Gillies, S., Culbertson, L., Bartos, M., Eubank, N., maxalbert, Bilogur,
203 A., ... Leblanc, F. (2020). *Geopandas/geopandas: v0.8.1* (Version v0.8.1). Zenodo.
204 <https://doi.org/10.5281/zenodo.3946761>
- 205 Khurelbaatar, G., Al Marzuqi, B., Van Afferden, M., Müller, R. A., & Friesen, J. (2021).
206 Data Reduced Method for Cost Comparison of Wastewater Management Scenarios—Case
207 Study for Two Settlements in Jordan and Oman. *Frontiers in Environmental Science*, 9.
208 <https://doi.org/10.3389/fenvs.2021.626634>
- 209 Li, G., & Matthew, R. G. S. (1990). New approach for optimization of urban drainage systems.
210 *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))
- 212 Maurer, M., Scheidegger, A., & Herlyn, A. (2013). Quantifying costs and lengths of urban
213 drainage systems with a simple static sewer infrastructure model. *Urban Water Journal*,
214 10(4), 268–280. <https://doi.org/10.1080/1573062X.2012.731072>
- 215 Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M.,
216 Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D.,
217 Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python.
218 *Journal of Machine Learning Research*, 12, 2825–2830.
- 219 Steele, J. C., Mahoney, K., Karovic, O., & Mays, L. W. (2016). Heuristic optimization model
220 for the optimal layout and pipe design of sewer systems. *Water Resources Management*,
221 30(5), 1605–1620. <https://doi.org/10.1007/s11269-015-1191-8>
- 222 Thomas, M., Trimble, T., & Sela, L. (2023). VisWaterNet: A python package for visualization

- 223 of water distribution networks. *Journal of Open Source Software*, 8(84), 5139. <https://doi.org/10.21105/joss.05139>
- 224 van Afferden, M., Cardona, J. A., Lee, M.-Y., Subah, A., & Müller, R. A. (2015). A new
225 approach to implementing decentralized wastewater treatment concepts. *Water Science
226 and Technology*, 72(11), 1923–1930. <https://doi.org/10.2166/wst.2015.393>
- 227 Water, U. (2018). *Sustainable Development Goal 6: Synthesis report 2018 on water and
228 sanitation* (United Nations Publications). United Nations. ISBN: 978-92-1-101370-2
- 229

DRAFT