

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Published: unpublished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 Pysewer is a tool for data-scarce environments using only few data and global assumptions –
³¹ thus enabling a transferability to a wide range of different regions. At the same time, a priori
³² data sources can be substituted with high-resolution data and site-specific information such as
³³ local water consumption and population data. The generated networks can then be exported
³⁴ (i.e., as a geopackage or shapefile) in order to utilise the results in preliminary planning stages,
³⁵ initial cost estimations, scenario development processes or for further comparison to decentral
³⁶ solutions where the network can be modified. The option to include several treatment locations
³⁷ also enables users to already plan decentralised networks or favour treatment locations (i.e.,
³⁸ due to local demands or restrictions).

39 Functionality and key features

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

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

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

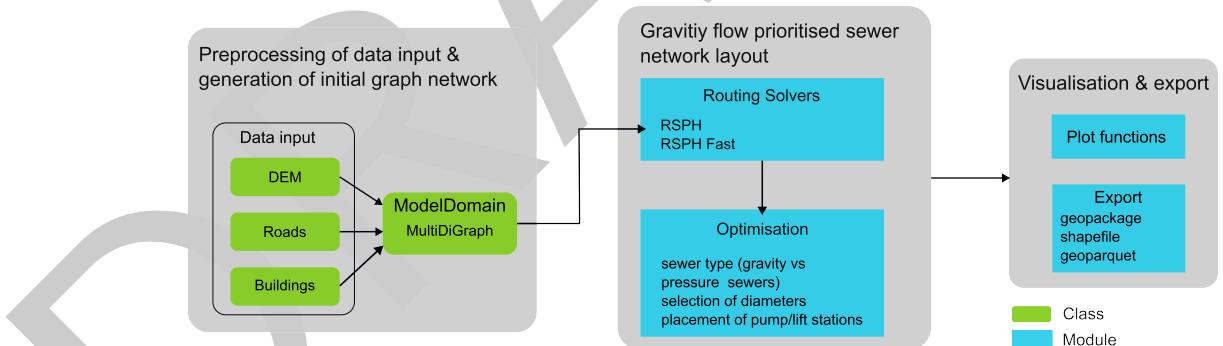


Figure 1: Pysewer's modular workflow

64 Preprocessing and initial network generation

65 In the preprocessing module, the roads, buildings and the DEM must all be projected in the
 66 same projection (CRS) and must be in the form of a geopandas ([Jordahl et al., 2020](#)) data
 67 frame or a shapefile. Roads, Buildings and DEM classes are used to transform the raw data
 68 formats into the required format (i.e., geopandas data frame) to create the initial graph network
 69 ([networkx, \(Hagberg et al., 2008\)](#)), where nodes represent crucial points such as junctions or
 70 buildings and edges to simulate potential sewer lines. The following measures ensure that the
 71 initial layout aligns with the road network and that there is serviceability to all buildings within
 72 the area of interest:

- 73 ▪ “connecting” buildings to the street network using the connect buildings method. This
 74 method adds nodes to the graph to connect the buildings in the network using the
 75 building points.

- 76 ▪ Creation of “virtual roads”. Buildings which are not directly connected to the road
 - 77 network are connected by finding the closest edge to the building, which is then marked
 - 78 as the closest edge. The nodes are then disconnected from the edges and are added to
 - 79 the initial connection graph network.
 - 80 ▪ Contracting the street network for more efficient graph traversal.
 - 81 ▪ Setting of the collection point or Wastewater Treatment Plant (WWTP). By default,
 - 82 the lowest elevation point in the region of interest is set as the location of the WWTP.
 - 83 Users can manually define the location of the WWTP by using the `add_sink` method.
- 84 After preprocessing, all relevant data is stored as a `MultiDiGraph` to allow for asymmetric edge
- 85 values (e.g., elevation profile and subsequently costs). [Figure 2](#) demonstrates the required
- 86 data, its preprocessing and the generation of the initial graph network.

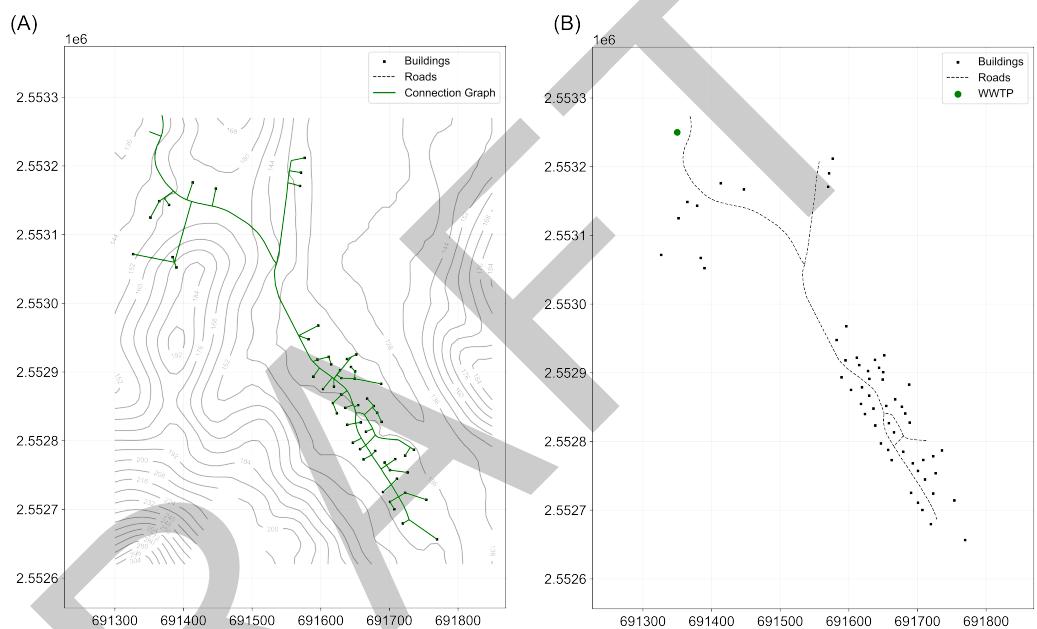


Figure 2: Pysewer preprocessing. Topographic map with the connection graph resulting from the instantiation of the `ModelDomain` class (A). Sewer network layout requirements: existing building, roads, and collection point (WWTP) (B).

87 Generating a gravity flow-prioritise sewer network

88 Within the computational framework of pysewer, the routing and optimisation modules function
 89 as the principal mechanisms for synthesising the sewer network. The objective of the routing
 90 module is to identify the paths through the network, starting from the sink. The algorithm
 91 approximates the directed Steiner tree (the Steiner arborescence) between all sources and the
 92 sink by using a repeated shortest path heuristic (RSPH). The routing module has two solvers
 93 to find estimates for the underlying minimum Steiner arborescence tree problem; these are:

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

99 In a nutshell, these solvers work by navigating through the connection graph (created using the
 100 `generate_connection_graph` method of the preprocessing module). This method simplifies
 101 the connection graph, removes any self-loops, sets trench depth node attributes to 0, and
 102 calculates the geometry, distance, profile, whether a pump is needed weight, and elevation

103 attributes for each edge and node. The shortest path between the subgraph and terminal
104 nodes in the connection graph is found using Dijkstra's Shortest Path Algorithm. The RSPH
105 solver repeatedly finds the shortest path between the subgraph nodes and the closest terminal
106 node, adding the path to the sewer graph and updating the subgraph nodes and terminal
107 nodes. Terminal nodes refer to the nodes in the connection graph that need to be connected
108 to the sink. On the other hand, subgraph nodes are the nodes in the directed routed Steiner
109 tree. These are initially set to the sink nodes and are updated as the RSPH solver is applied
110 to find the shortest path between the subgraph and the terminal nodes. This way, all terminal
111 nodes are eventually connected to the sink.

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

- 117 1. Terrain does not allow for gravity flow to the downstream node (this check uses the
118 needs_pump attribute from the preprocessing to reduce computational load)—placement
119 of a pump station is required.
- 120 2. Terrain does not require a pump, but the lowest inflow trench depth is too low for
121 gravitational flow—placement of a lift station is required.

122 Gravity flow is possible within given constraints—the minimum slope is achieved, no pump
123 or lifting station is required. As our tool strongly focuses on prioritising gravity flow, a high
124 pump penalty is applied to minimise the length of the pressure sewers. The pumping penalty
125 expressed as the edge weight is relative to the trench depth required to achieve minimum slope
126 to achieve self-cleaning velocities in a gravity sewer. The maximum trench depth t_{\max} required
127 to achieve the minimum slope is set at $t_{\max} = 8$ in the default settings of pysewer. When
128 there is a need to dig deeper than this predefined value, then a pump is required.

129 The optimisation module also facilitates the selection of the diameters to be used in the
130 network and peak flow estimation, as well as the key sewer attributes such as the number of
131 pump or lifting stations, the length of pressure and gravity sewers, which can be visualised
132 and exported for further analysis. [Figure 3](#) shows an example of a final sewer network layout
133 generated after running the calculation of the hydraulics parameters.

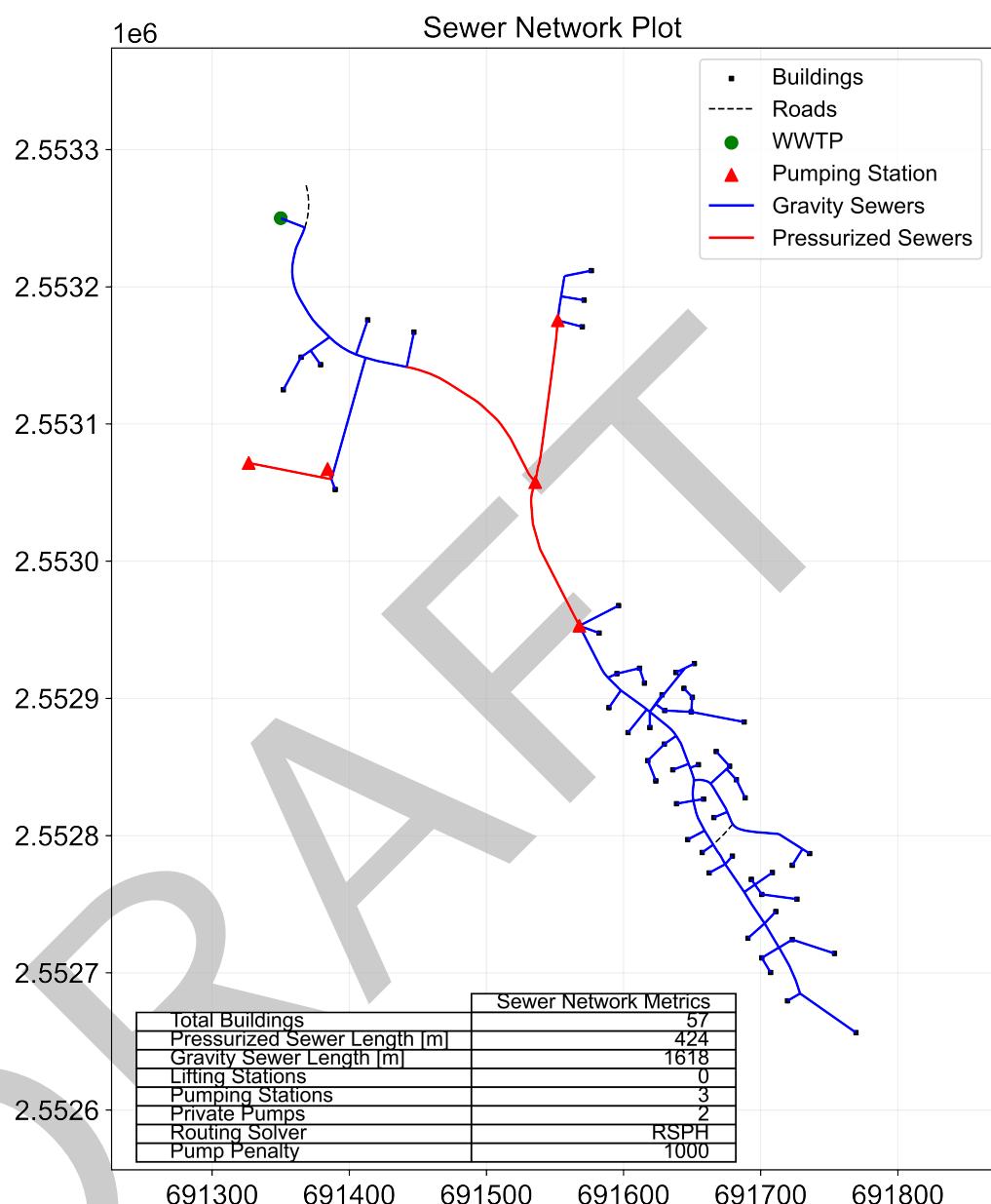


Figure 3: Pysewer optimisation. Final layout of the sewer network.

134 Visualising and exporting the generated sewer network

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

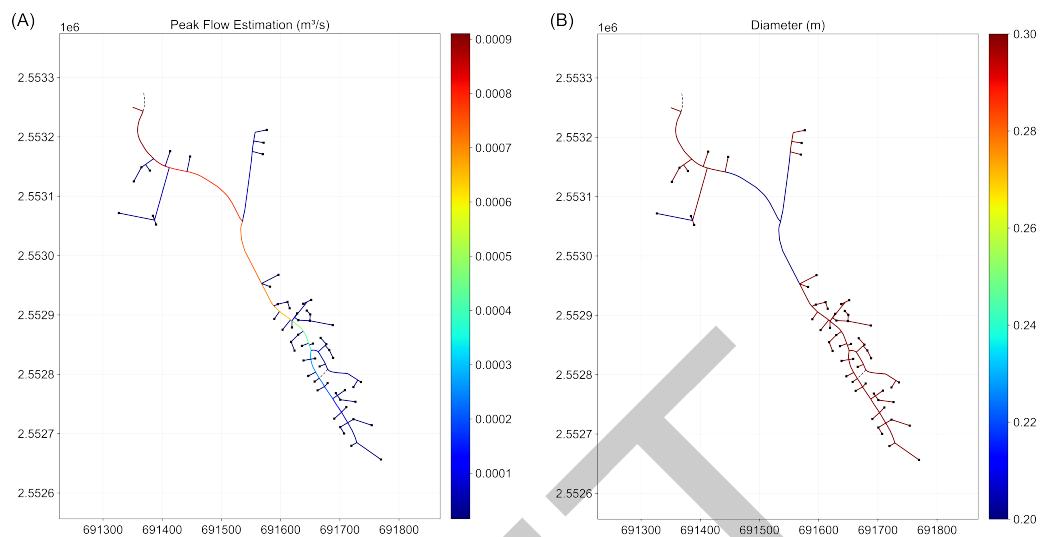


Figure 4: Pysewer visualisation. Attributes of the sewer network layout. Peak flow estimation (A), Pipe diameters selected (B)

Acknowledgement

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

Software citations

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

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

References

- GDAL/OGR contributors. (2023). *GDAL/OGR geospatial data abstraction software library*. Open Source Geospatial Foundation. <https://doi.org/10.5281/zenodo.5884351>
- Hagberg, A. A., Schult, D. A., & Swart, P. J. (2008). Exploring network structure, dynamics, and function using NetworkX. In G. Varoquaux, T. Vaught, & J. Millman (Eds.), *Proceedings of the 7th python in science conference* (pp. 11–15).
- Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Haldane, A., Río, J. F. del, Wiebe, M., Peterson, P., ... Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, 585(7825), 357–362. <https://doi.org/10.1038/s41586-020-2649-2>

- 167 Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science &*
168 *Engineering*, 9(3), 90–95. <https://doi.org/10.1109/MCSE.2007.55>
- 169 Jordahl, K., Bossche, J. V. den, Fleischmann, M., Wasserman, J., McBride, J., Gerard,
170 J., Tratner, J., Perry, M., Badaracco, A. G., Farmer, C., Hjelle, G. A., Snow, A. D.,
171 Cochran, M., Gillies, S., Culbertson, L., Bartos, M., Eubank, N., maxalbert, Bilogur,
172 A., ... Leblanc, F. (2020). *Geopandas/geopandas: v0.8.1* (Version v0.8.1). Zenodo.
173 <https://doi.org/10.5281/zenodo.3946761>
- 174 Khurelbaatar, G., Al Marzuqi, B., Van Afferden, M., Müller, R. A., & Friesen, J. (2021).
175 Data Reduced Method for Cost Comparison of Wastewater Management Scenarios—Case
176 Study for Two Settlements in Jordan and Oman. *Frontiers in Environmental Science*, 9.
177 <https://doi.org/10.3389/fenvs.2021.626634>
- 178 Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M.,
179 Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D.,
180 Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python.
181 *Journal of Machine Learning Research*, 12, 2825–2830.
- 182 van Afferden, M., Cardona, J. A., Lee, M.-Y., Subah, A., & Müller, R. A. (2015). A new
183 approach to implementing decentralized wastewater treatment concepts. *Water Science
184 and Technology*, 72(11), 1923–1930. <https://doi.org/10.2166/wst.2015.393>
- 185 Water, U. (2018). *Sustainable Development Goal 6: Synthesis report 2018 on water and
186 sanitation* (United Nations Publications). United Nations. ISBN: 978-92-1-101370-2

DRAFT