The Journal of Open Source Software

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

Moritz Sanne[1,2], **Ganbaatar Khurelbaatar**[1], **Daneish Despot**[1¶], **Manfred van Afferden**[1], **and Jan Friesen**[1]

1 Centre for Environmental Biotechnology, Helmholtz Centre for Environmental Research GmbH – UFZ, Permoserstraße 15 | 04318 Leipzig, Germany **2** Europace AG, Berlin, Germany ¶ Corresponding author

DOI: 10.21105/joss.06430

Software

- Review
- Repository
- Archive

Editor: Olexandr Konovalov

Reviewers:

- @meghnathomas
- @barneydobson

Submitted: 25 October 2023
Published: 10 December 2024

Summary

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

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

Statement of need

The sustainable management of water and sanitation has been defined as one of the UN's sustainable development goals: SDG 6 (UN-Water, 2018). As of 2019, SDG 6 might not be reached in 2030 despite the progress made, which means that more than half of the population still lacks safely managed sanitation (UN-Water, 2018).
In order to identify optimal wastewater management at the settlement level, it is necessary to compare different central or decentral solutions. To achieve this, a baseline is required against which other scenarios can be compared (Khurelbaatar et al., 2021; van Afferden et al., 2015). To this end, we developed pysewer – a tool that generates settlement-wide sewer networks, which connect all the buildings within the settlement boundary or the region of interest to one or more wastewater treatment plant locations.

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

Pysewer is designed for data-scarce environments, utilizing only minimal data and global 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).

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.

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

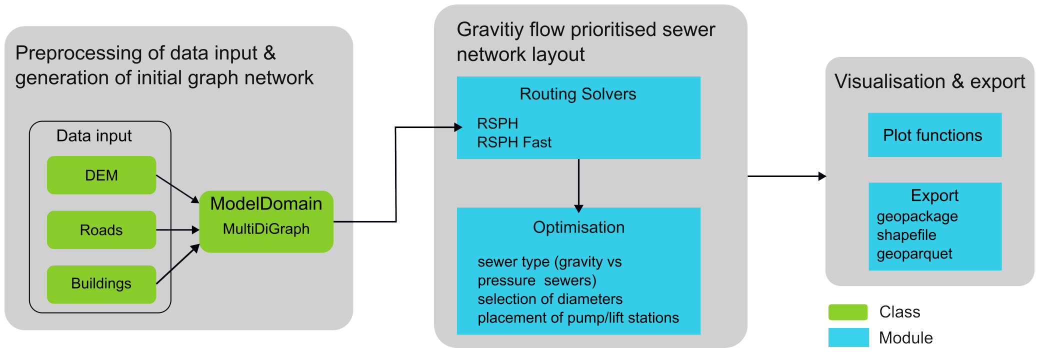


Figure 1: Pysewer's modular workflow

Preprocessing and initial network generation

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

- "Connecting" buildings to the street network using the connect buildings method. This method adds nodes to the graph to connect the buildings in the network using the building points.
- Creation of "virtual roads". Buildings which are not directly connected to the road network are connected by finding the closest edge to the building, which is then marked as the closest edge. The nodes are then disconnected from the edges and are added to the initial connection graph network.
- Simplifying the street network for more efficient graph traversal.
- Setting of the collection point or Wastewater Treatment Plant (WWTP). By default, the lowest elevation point in the region of interest is set as the location(s) of the WWTP. Users can manually define the location of the WWTP by using the `add_sink` method.

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

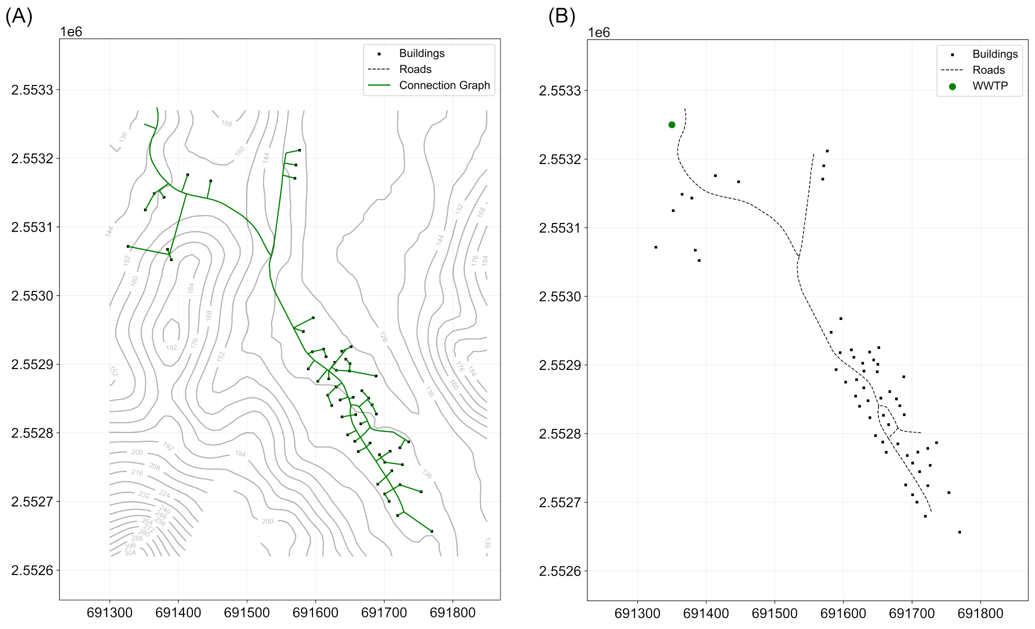


Figure 2: Pysewer preprocessing. Topographic map with the connection graph resulting from the instantiation of the `ModelDomain` class (A). Sewer network layout requirements: existing building, roads, and collection point (WWTP) (B).

Generating a gravity flow-prioritise sewer network

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

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.
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.

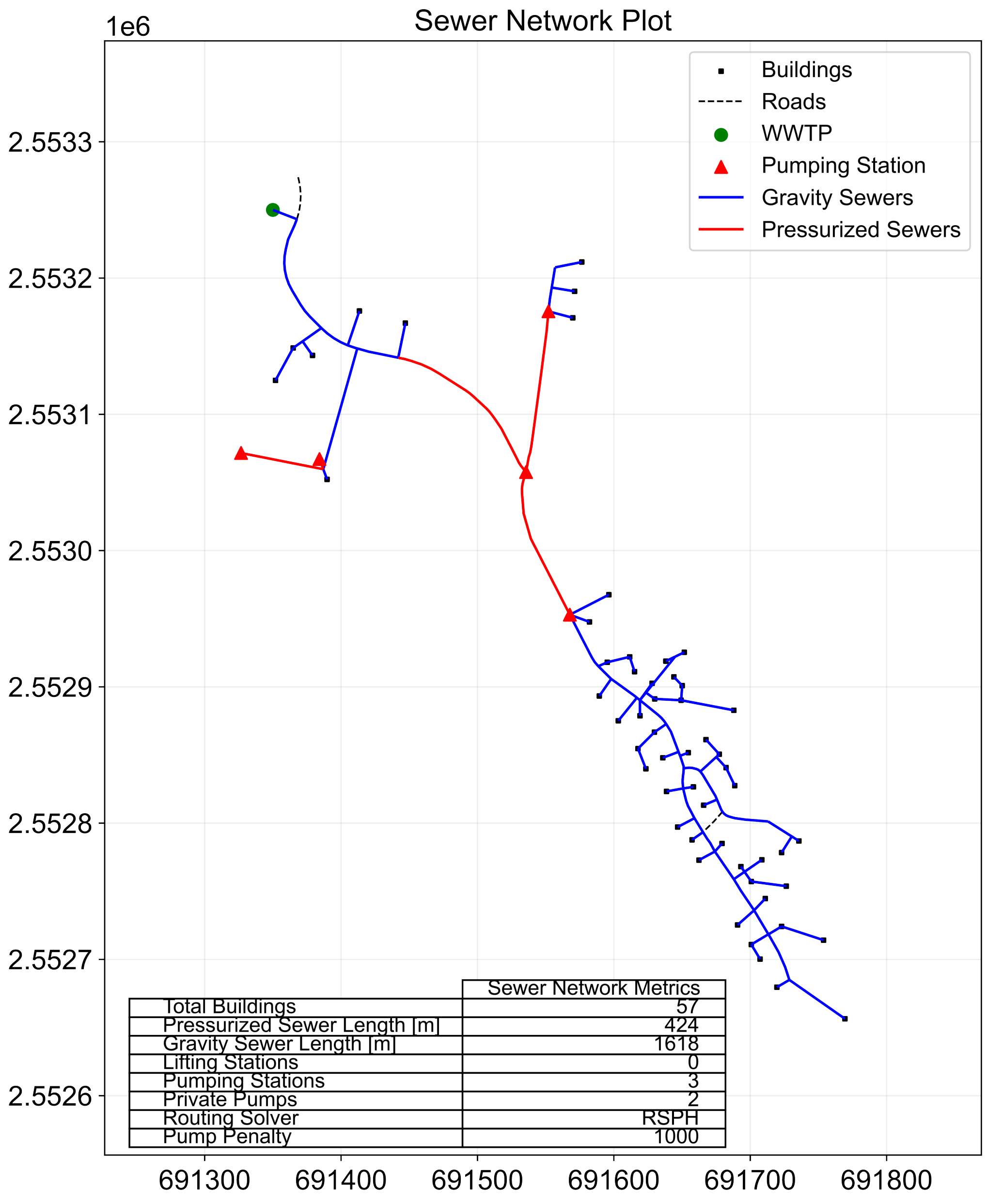
In a nutshell, these solvers work by navigating through the connection graph (created using the `generate_connection_graph` method of the preprocessing module). This method first simplifies the connection graph by removing any self-loops and setting trench depth node attributes to 0. It then calculates key parameters such as geometry, distance, profile, initial edge weights (needed for placing pump stations), and elevation attributes for each edge and node. The shortest path between the subgraph and terminal 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 the closest terminal node, adding the path to the sewer graph and updating the subgraph nodes and terminal nodes. Terminal nodes refer to the nodes in the connection graph that need to be connected to the sink. On the other hand, subgraph nodes are the nodes in the directed routed Steiner tree. These are initially set to the sink nodes and are updated as the RSPH solver is applied to find the shortest path between the subgraph and the terminal nodes. This way, all terminal nodes are eventually connected to the sink.

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

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

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

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



	Sewer Network Metrics
Total Buildings	57
Pressurized Sewer Length [m]	424
Gravity Sewer Length [m]	1618
Lifting Stations	0
Pumping Stations	3
Private Pumps	2
Routing Solver	RSPH
Pump Penalty	1000

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

Visualising and exporting the generated sewer network

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

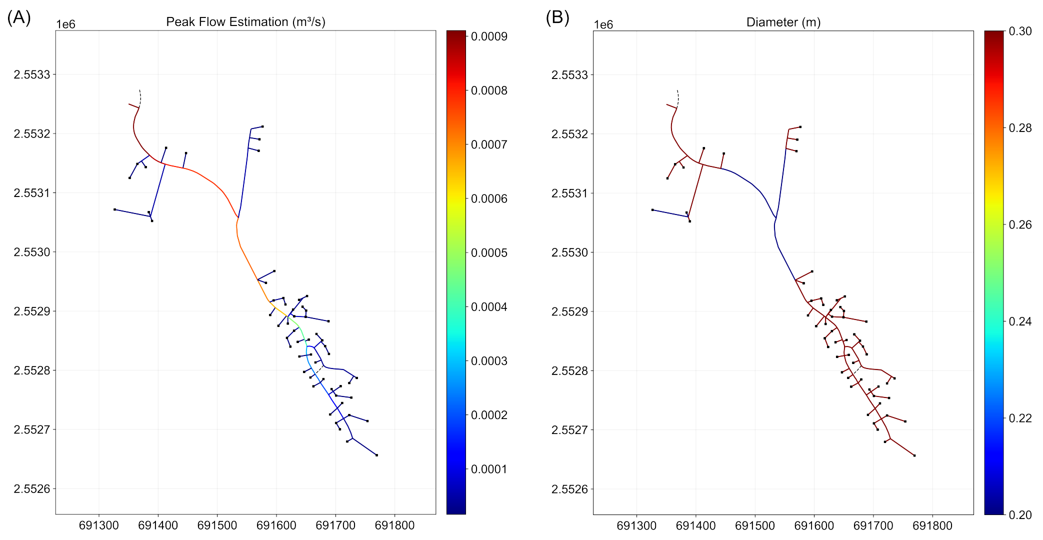


Figure 4: Pysewer visualisation. Attributes of the sewer network layout. Peak flow estimation (A), Pipe diameters selected (B)

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

- Geopandas 0.8.1 (Jordahl et al., 2020)
- NetworkX 3.1 (Hagberg et al., 2008)
- Rasterio 1.2.10 (Gillies & others, 2021)
- Numpy 1.25.2 (Harris et al., 2020)
- Matplotlib 3.7.1 (Hunter, 2007)
- Scikit-learn 1.0.2 (Pedregosa et al., 2011)
- GDAL 3.0.2 (GDAL/OGR contributors, 2019)

Author contributions

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

References

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

Dijkstra, E. W. (1959). A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik*, *1*(1), 269–271. https://doi.org/10.1007/BF01386390

Duque, N., Duque, D., Aguilar, A., & Saldarriaga, J. (2020). Sewer Network Layout Selection and Hydraulic Design Using a Mathematical Optimization Framework. *Water*, *12*(12), Article 12. https://doi.org/10.3390/w12123337

Friesen, J., Sanne, M., Khurelbaatar, G., & Afferden, M. van. (2023). “OCTOPUS” principle reduces wastewater management costs through network optimization and clustering. *One Earth*, *6*(9), 1227–1234. https://doi.org/10.1016/j.oneear.2023.08.005

GDAL/OGR contributors. (2019). *GDAL/OGR Geospatial Data Abstraction software Library* (Version v3.0.2) [Computer software]. Open Source Geospatial Foundation. https://doi.org/10.5281/zenodo.5884351

Gillies, S., & others. (2021). *Rasterio: Geospatial raster I/O for Python programmers* (Version v1.2.10). Mapbox. https://github.com/rasterio/rasterio

Hagberg, A. A., Schult, D. A., & Swart, P. J. (2008). Exploring Network Structure, Dynamics, and Function using NetworkX. *Scipy*. https://doi.org/10.25080/TCWV9851

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

Holland, M. E. (1966). *Computer Models of Waste-Water Collection Systems* [PhD thesis]. Harvard University.

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Hwang, F. K., & Richards, D. S. (1992). Steiner Tree Problems. *Networks*, *22*(1), 55–89. https://doi.org/10.1002/net.3230220105

Jordahl, K., Bossche, J. V. den, Fleischmann, M., Wasserman, J., McBride, J., Gerard, J., Tratner, J., Perry, M., Badaracco, A. G., Farmer, C., Hjelle, G. A., Snow, A. D., Cochran, M., Gillies, S., Culbertson, L., Bartos, M., Eubank, N., maxalbert, Bilogur, A., … Leblanc, F. (2020). *Geopandas* (Version v0.8.1). Zenodo. https://doi.org/10.5281/zenodo.3946761

Khurelbaatar, G., Al Marzuqi, B., Van Afferden, M., Müller, R. A., & Friesen, J. (2021). Data Reduced Method for Cost Comparison of Wastewater Management Scenarios – Case Study for Two Settlements in Jordan and Oman. *Frontiers in Environmental Science*, *9*. https://doi.org/10.3389/fenvs.2021.626634

Li, G., & Matthew, R. G. S. (1990). New Approach for Optimization of Urban Drainage Systems. *Journal of Environmental Engineering*, *116*(5), 927–944. https://doi.org/10.1061/(ASCE)0733-9372(1990)116:5(927)

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

Momeni, A., Chauhan, V., Bin Mahmoud, A., Piratla, K. R., & Safro, I. (2023). Generation of Synthetic Water Distribution Data Using a Multiscale Generator-Optimizer. *Journal of Pipeline Systems Engineering and Practice*, *14*(1). https://doi.org/10.1061/jpsea2.pseng-1358

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, Édouard. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*(85), 2825–2830. http://jmlr.org/papers/v12/pedregosa11a.html

Steele, J. C., Mahoney, K., Karovic, O., & Mays, L. W. (2016). Heuristic Optimization Model for the Optimal Layout and Pipe Design of Sewer Systems. *Water Resources Management*, *30*(5), 1605–1620. https://doi.org/10.1007/s11269-015-1191-8

UN-Water. (2018). *Sustainable Development Goal 6: Synthesis Report 2018 on Water and Sanitation* (United Nations Publications). United Nations. ISBN: 978-92-1-101370-2

van Afferden, M., Cardona, J. A., Lee, M.-Y., Subah, A., & Müller, R. A. (2015). A New Approach to Implementing Decentralized Wastewater Treatment Concepts. *Water Science and Technology*, *72*(11), 1923–1930. https://doi.org/10.2166/wst.2015.393