NORWEGIAN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Faculty of Engineering

Department of Ocean Operations and Civil Engineering

Modeling and Simulation of Urban Stormwater Collection Systems using Integration of SWMM, GIS, and Python

Tutorial 1

Razak Seidu

Lam Van Nguyen

Water and Environmental Engineering Group

NTNU Ålesund

07/2022

Objectives of this tutorial

The overall aim of this tutorial is to provide a user with the way to use SWMM for modeling and simulating stormwater collection systems. The tutorial also provides a step-by-step guideline for building an SWMM project using GIS and Python. An open-source package (PySWMM) is also introduced in this tutorial helping run an SWMM project automatically. Essential steps for automatic SWMM modeling are also provided in this tutorial.

Table of Contents

1. Introduction

The EPA Storm Water Management Model (SWMM) is a dynamic rainfall-runoff simulation model used for single event or long-term (continuous) simulation of runoff quantity and quality from primarily urban areas. SWMM is open-source software and is developed by the Water Supply and Water Resources Division of the U.S. Environmental Protection Agency's National Risk Management Research Laboratory. Users who are interested in this software can download it at: https://www.epa.gov/water-research/storm-water-management-model-swmm.

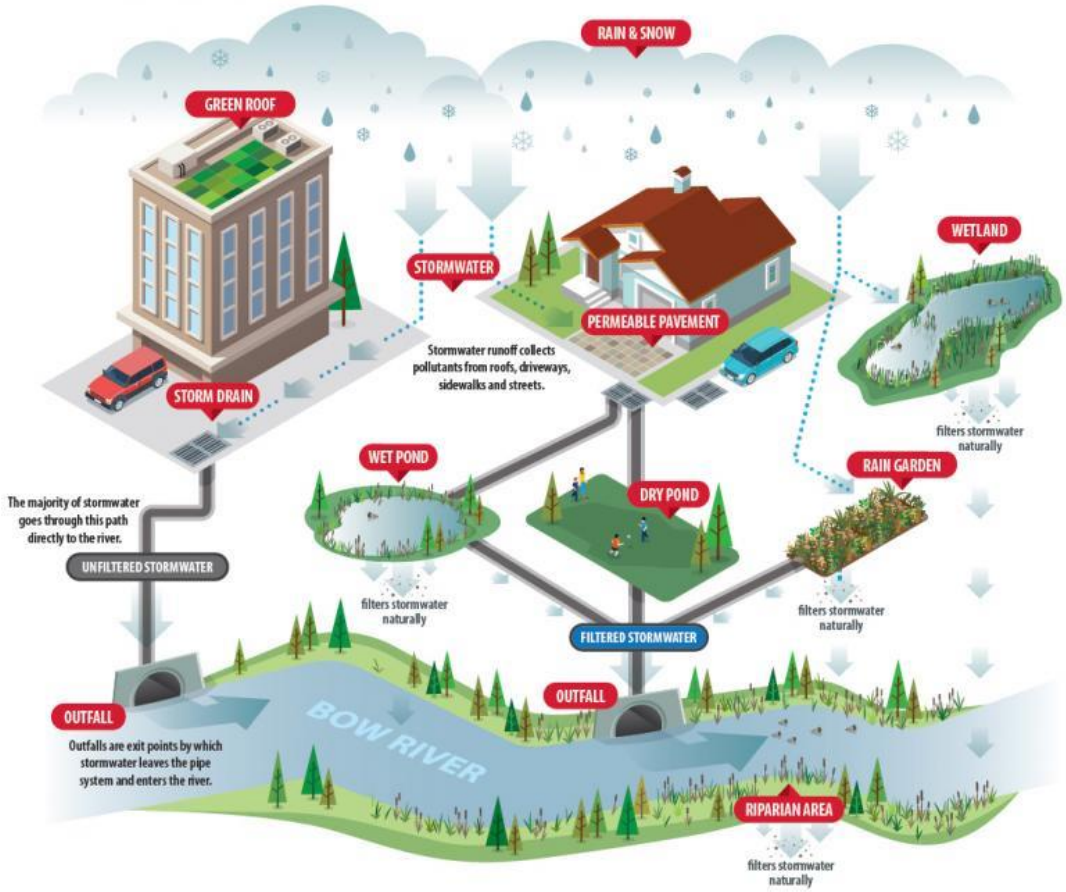


Figure 1-1. Stormwater System in the context of Green Infrastructure

In this tutorial, we do not deeply explain the elements of equations. Users/readers can refer to academic materials to have an overview. Default parameters will be prioritized in setting up the SWMM model.

1.1. Main Applications of SWMM

- Design and sizing of drainage system components for flood control
- Sizing of detention facilities and their appurtenances for flood control and water quality protection

- Flood plain mapping of natural channel systems
- Designing control strategies for minimizing combined sewer overflows
- Evaluating the impact of inflow and infiltration on sanitary sewer overflows
- Generating non-point source pollutant loadings for waste load allocation studies
- Evaluating the effectiveness of Best Management Practices (BMPs) for reducing wet weather pollutant loadings

1.2. SWMM Installation

In this tutorial, related software will be installed on the Windows Operating System computer. Installation steps on other operating systems (such as macOS, Linus, etc.,) can be slightly different.

Installing SWMM includes several main steps:

- *Step 1:* Visit the home page of the U.S. Environmental Protection Agency by clicking on the above-provided link
- *Step 2:* Download the latest SWMM installation package from the webpage depending on the user's operation system (32-bit or 64-bit). The latest SWMM version when writing this tutorial is 5.2.0 (Figure 1-2).

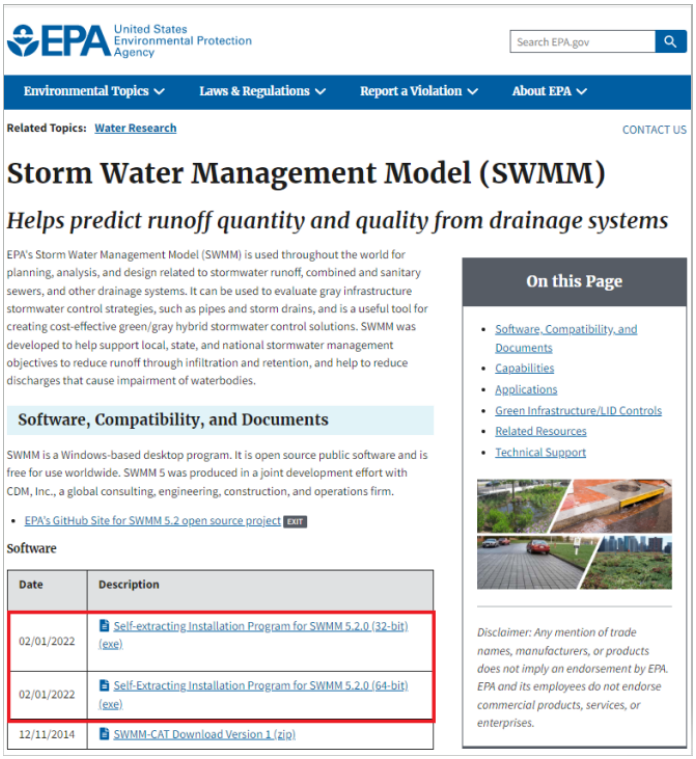


Figure 1-2. SWMM downloading homepage

- *Step 3:* Store the downloaded SWMM package on a personal computer (PC) or hard disk. Install SWMM by double click on the downloaded package.
- *Step 4:* Select the location for SWMM installation. Hit the *"Next"* button in some next steps to finish the installation (Figure 1-3).

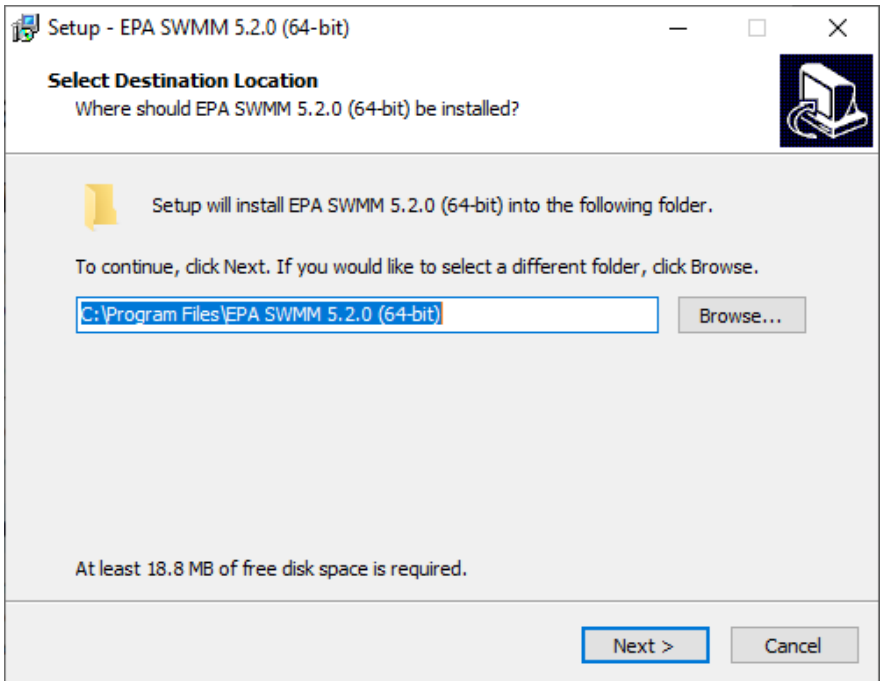


Figure 1-3. SWMM installation

- *Step 5:* Finish installation and start SWMM (Figure 1-4).

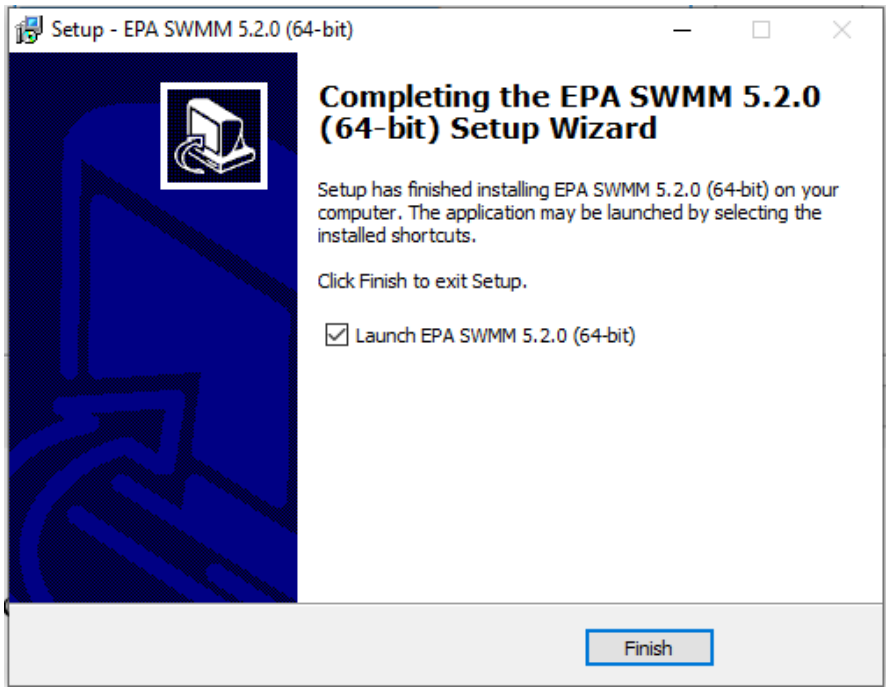


Figure 1-4. Finishing SWMM installation

➢ *Step 6:* Start an empty SWMM project (Figure 1-5).

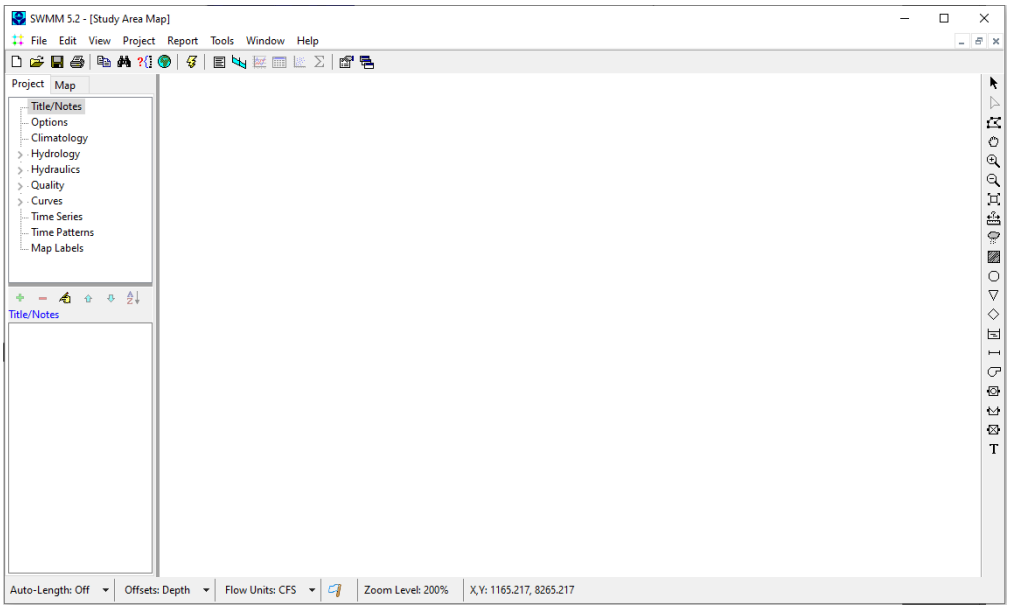


Figure 1-5. Starting an empty SWMM project

1.3. QGIS Installation

QGIS is one of the most widely used Geographic Information System (GIS) software, likely ArcMap, or ArcGIS Pro. QGIS is a free and open-source cross-platform desktop GIS application that supports viewing, editing, printing, and analysis of geospatial data.

To download QGIS, the user can follow the below steps:

➢ *Step 1:* Visit this link: https://www.qgis.org/en/site/ and download QGIS software (Figure 1-6). Documentations for using QGIS can be found at this address.

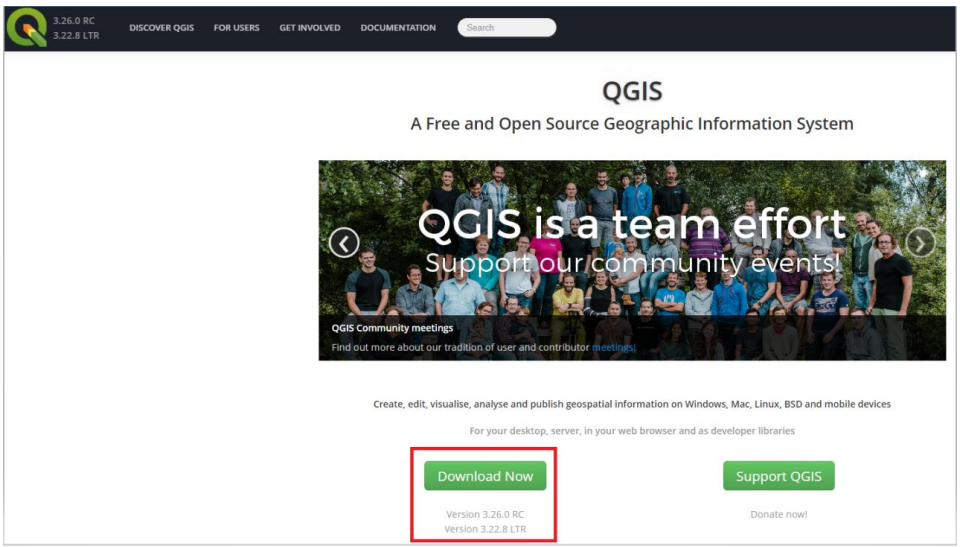


Figure 1-6. QGIS downloading homepage

- *Step 2:* Download the correct QGIS version depending on the user's operating system (Figure 1-7).

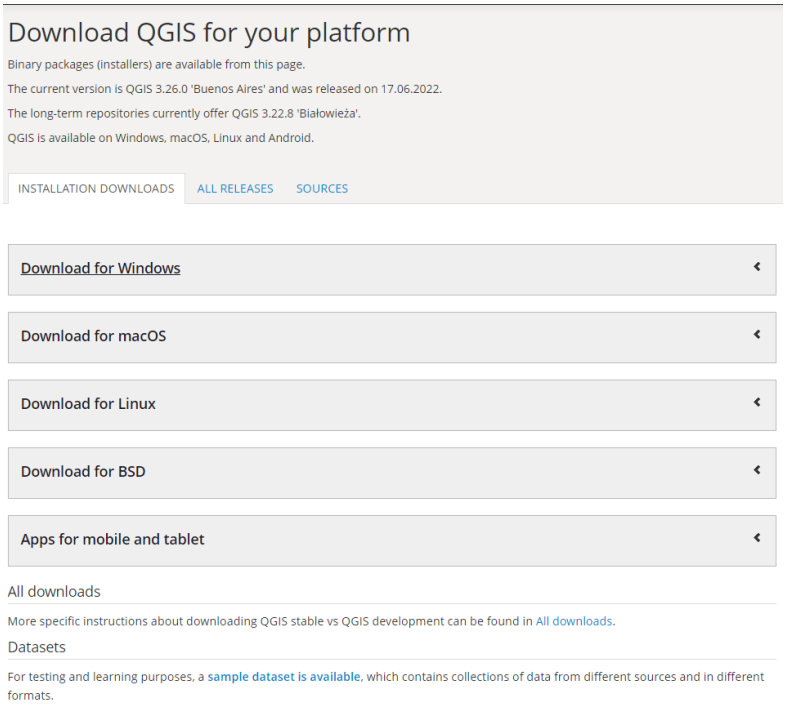


Figure 1-7. Multi-platform QGIS

- *Step 3:* Select to download the QGIS. In this tutorial, we used QGIS version 3.22 (the most stable long-term release version) (Figure 1-8).

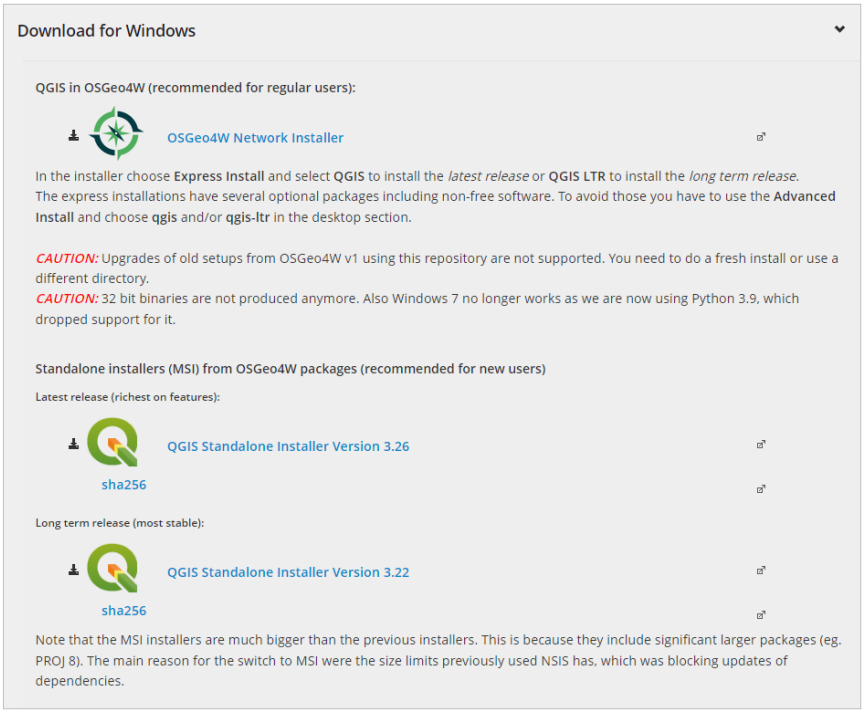


Figure 1-8. Selecting the wanted QGIS version to download

- *Step 4:* Store the downloaded QGIS software on the PC or hard disk. Install QGIS by double click on the downloaded item.
- *Step 5:* Complete the installation process by clicking the *"Next"* button in the next steps (Figure 1-9).

Figure 1-9. QGIS installation process

- *Step 6:* Finish installation and start a new QGIS project (Figure 1-10).

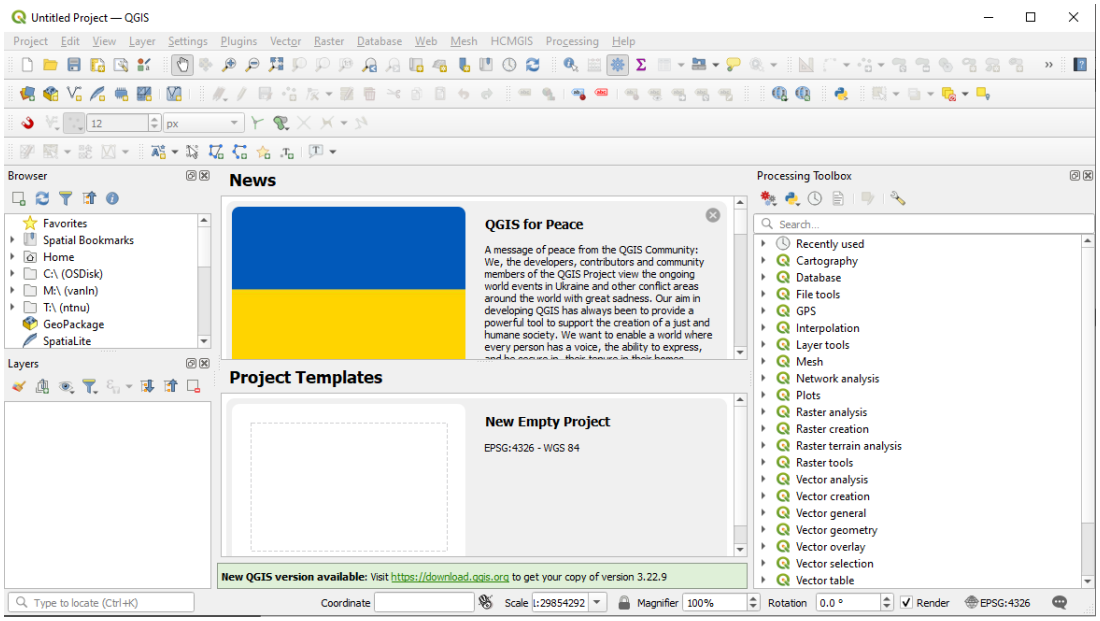


Figure 1-10. Opening a new QGIS project

2. Data Preparation

2.1. Drainage Network Structure in SWMM

Basic components of a collection network consist of junctions, conduits, sub-catchments, rain gage, outfalls, regulations, pumps, storage units, etc., The relationship between these components is shown in Figure 2-1.

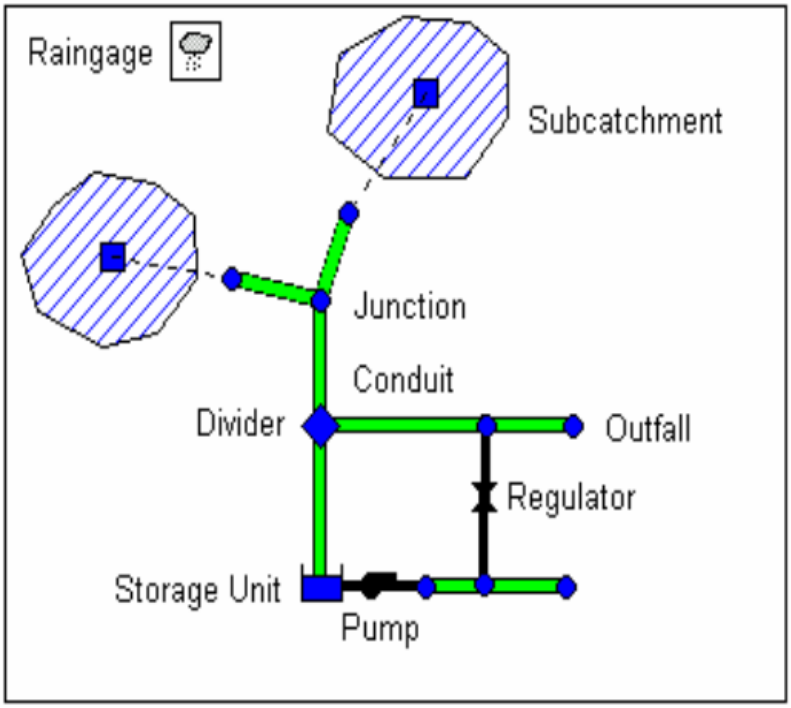


Figure 2-1. Basic components in a collection network system

Some basic steps for simulating SWMM include:

- *Step 1:* Set up default parameters for specific components (junctions, conduits, sub-catchments, etc.,)
- *Step 3:* Design collection network system
- *Step 4:* Run simulation
- *Step 5:* Check results
- *Step 6:* Adjust input parameters and re-run simulation (if necessary)

2.2. Setting up Default Parameters

To set up initial parameters for an SWMM project, a new project must be created. To create a new SWMM project, the user can click on the symbol  on the desktop or look for it from the start menu on the PC. A blank SWMM project will appear (Figure 2-2).

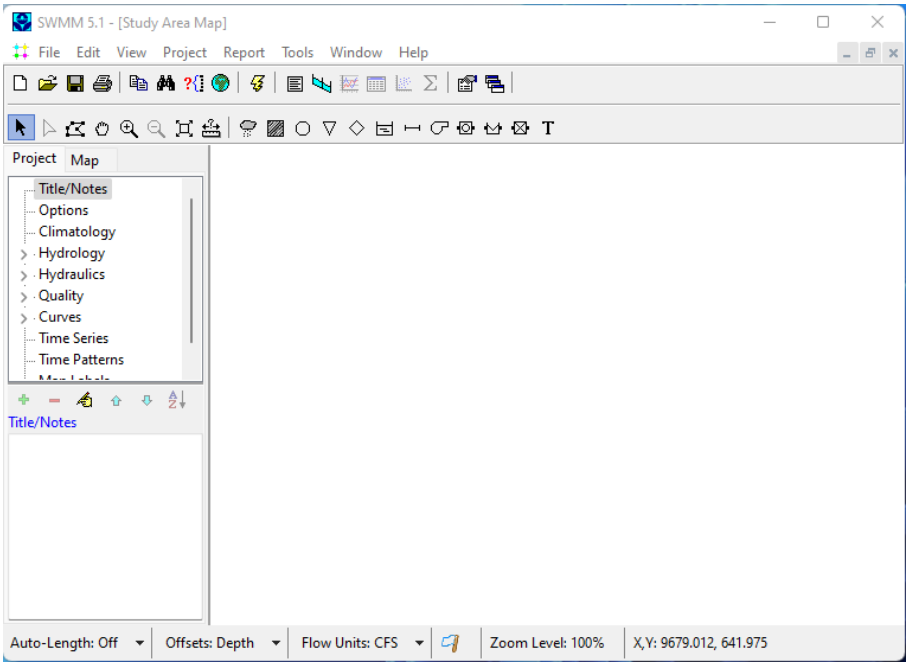


Figure 2-2. Starting an SWMM project

To set up parameters for the SWMM project: click ***Option*** on the left panel of SWMM software (Figure 2-3).

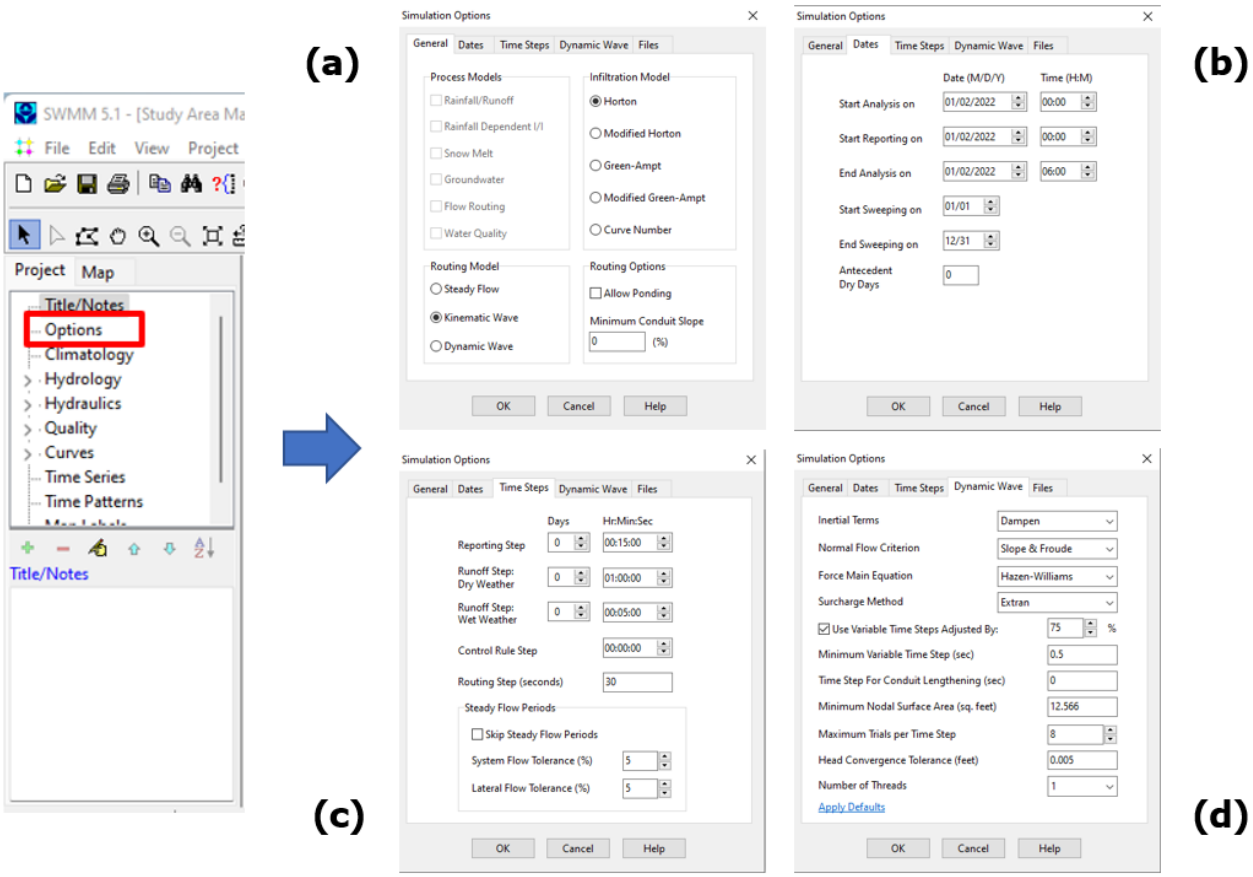


Figure 2-3. Setting up default parameters for the SWMM project

A *"Simulation Options"* window will appear likely in Figure 2-3. In this window, the user can:

- Select the routing models or change the default infiltration models (Figure 2-3a)
- Change simulation period (Figure 2-3b)
- Adjust time step for simulation report or weather-related parameters (Figure 2-3c)
- Set up other parameters (Figure 2-3d)

Default parameters for SWMM basic components (such as junctions, conduits, and sub-catchments) can be specified by selecting the *"Project"* option on the main menu and clicking the *"Defaults..."* option (Figure 2-4).

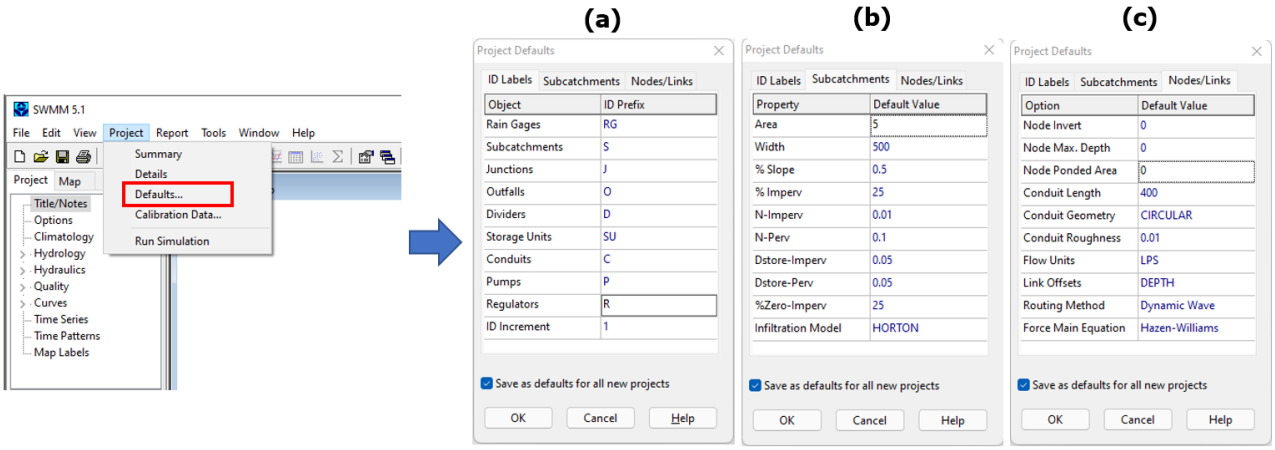


Figure 2-4. Setting up default parameters for components

The *"Project Defaults"* window allows the user to:

- Set up ID prefix for basic components in an SWMM project (Figure 2-4a)
- Specify important values for sub-catchments (Figure 2-4b)
- Adjust essential values for conduits and junctions (Figure 2-4c)

The atmosphere-related elements in SWMM are declared by using the climatological option. This window allows the user to set up time series-related data. For instance, the user can set up the change of temperature, and select/adjust the evaporation degree as well as wind speed and snow melt during the simulation period (Figure 2-5).

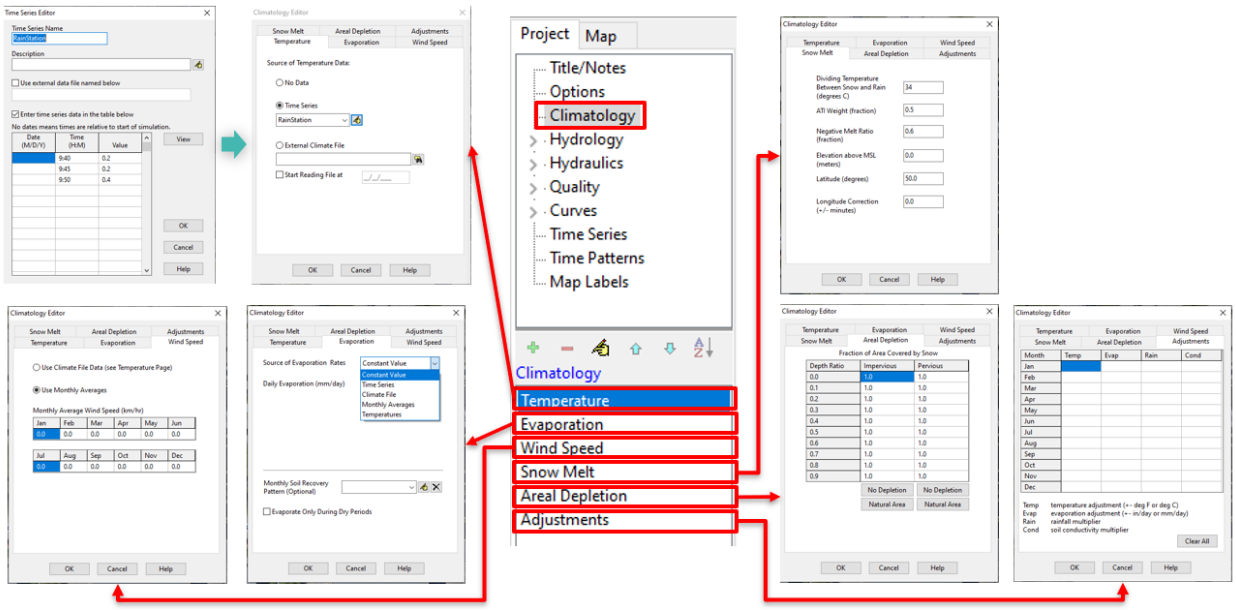


Figure 2-5. Climatological data in SWMM

2.3. Introduction to basic components in the SWMM project

An SWMM project has been built from basic components that are junctions, conduits, sub-catchments, rain gage, outfalls, regulations, pumps, storage units, etc., These components can be created and imported from the toolbar menu (Figure 2-6).

Figure 2-6. The toolbar menu in SWMM

The overview of basic components in SWMM is represented in Table 2-1.

Table 2-1. Basic components in SWMM

Component	Description	Symbol in SWMM
Rain Gage	These components supply precipitation data for one or more Sub-catchment areas	
Sub-catchment	They are hydrologic units of land whose topography and drainage system elements direct surface runoff to a single discharge point	
Junction	They are drainage system nodes where links join	
Outfall	They are terminal nodes of the drainage system used to define final downstream boundaries under Dynamic Wave flow routing	
Flow Divider	They are drainage system nodes that divert inflows to a specific conduit in a prescribed manner	

Storage Unit	They are drainage system nodes that provide storage volume	
Conduit	They are pipes or channels that move water from one node to another in the conveyance system	
Pump	They are links used to lift water to higher elevations	
Orifice	They are used to model outlet and diversion structures in drainage systems, which are typically openings in the wall of a manhole, storage facility, or control gate	
Weir	They are used to model outlet and diversion structures in a drainage system. Weirs are typically located in a manhole, along the side of a channel, or within a storage unit	
Outlet	They are flow control devices that are typically used to control outflows from storage units	

2.4. Setting up Time-series Data

Simulation in SWMM is a time-depending process. Therefore, the time-series input data can be created previously for simulation.

To create a new curve or edit exiting curves of components (such as pump, storage, etc.,), the user can use the curve editor function in SWMM by selecting the *"Curves"* option on the left panel of SWMM software (Figure 2-7a).

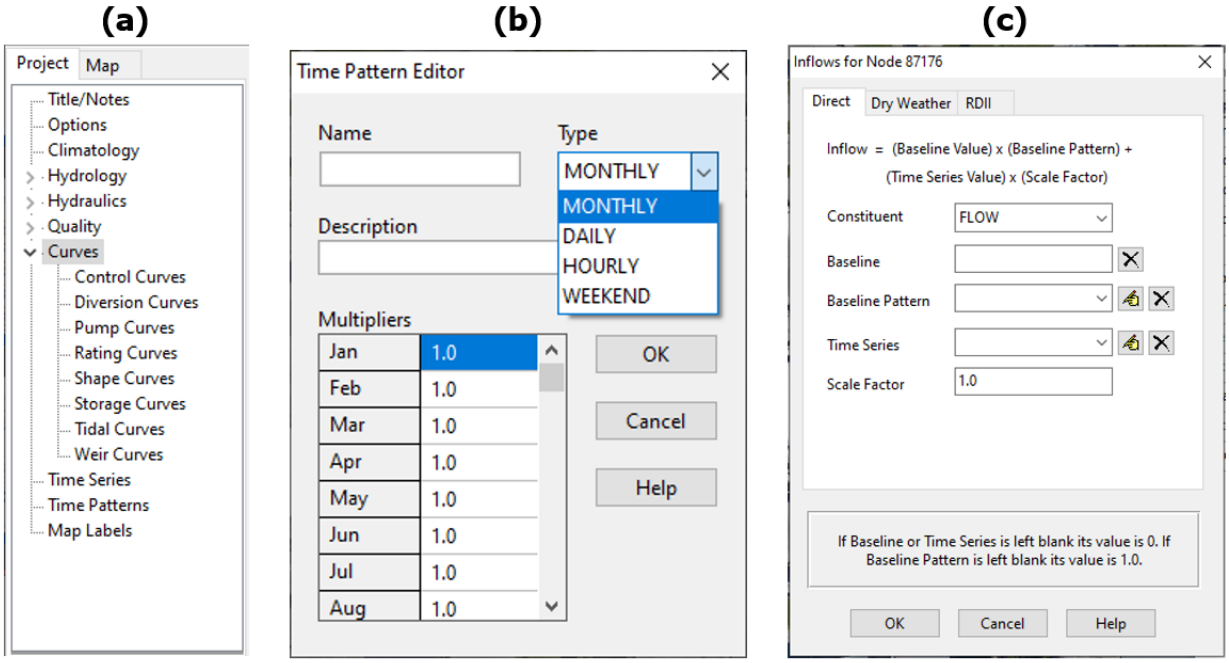


Figure 2-7. Curve editor window

The *"Time Pattern Editor"* window allows the user to create new time pattern objects or

edit existing time pattern objects. It is used to define pattern data for inflow into junction(s), a monthly pattern for N-Perv, Dstore, and Infiltration into Sub-catchment(s) (Figure 2-7b).

The *"Inflows for Node"* window allows the user to assign Direct, Dry Weather, and RDII inflow into a specific node/junction (Figure 2-7c).

2.5. Designing an SWMM project

a. Georeferencing Background Image

To support the design of a sewer network in SWMM, it is recommended to put this network on a real scale in the SWMM project. To do this, the background image used for designing the sewer network should be put and georeferenced in SWMM. In this tutorial, we will introduce the way to create a background image using QGIS and add geographic information to this image in SWMM.

❖ *Installing Plugin*

In QGIS, to displace the background image, some plugins (such as HCMGIS or QuickMapServices) can be installed. In this tutorial, we used the plugin HCMGIS. The process for installing a plugin is represented in Figure 2-8.

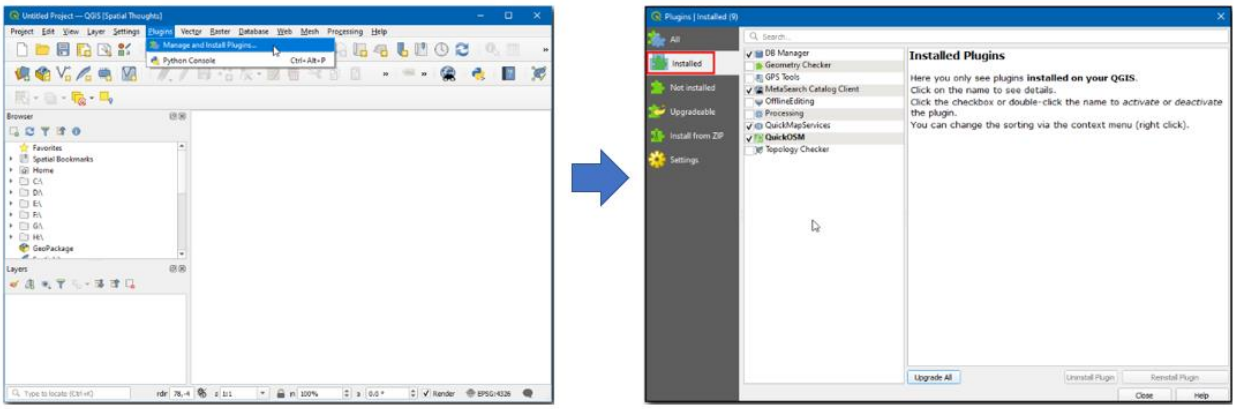


Figure 2-8. Installing a plugin in QGIS

Type the keyword *"HCMGIS"* (or *"QuickMapServices"*) into the search address to download and install HCMGIS (or QuickMapServices) plugin. To insert a background image into QGIS, select the wanted type of background image from the list (Figure 2-9).

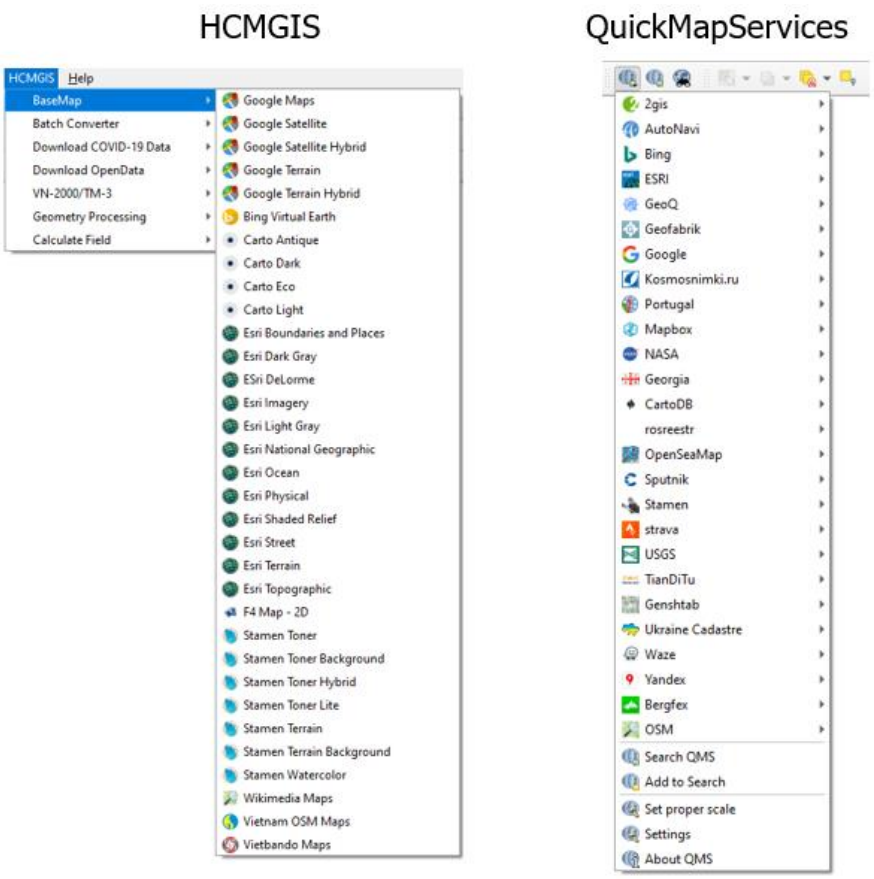


Figure 2-9. Background images in QGIS

❖ *Specifying the Study Area*

In QGIS, the user uses Zoom/Move functions to look for the interesting area (Figure 2-10).

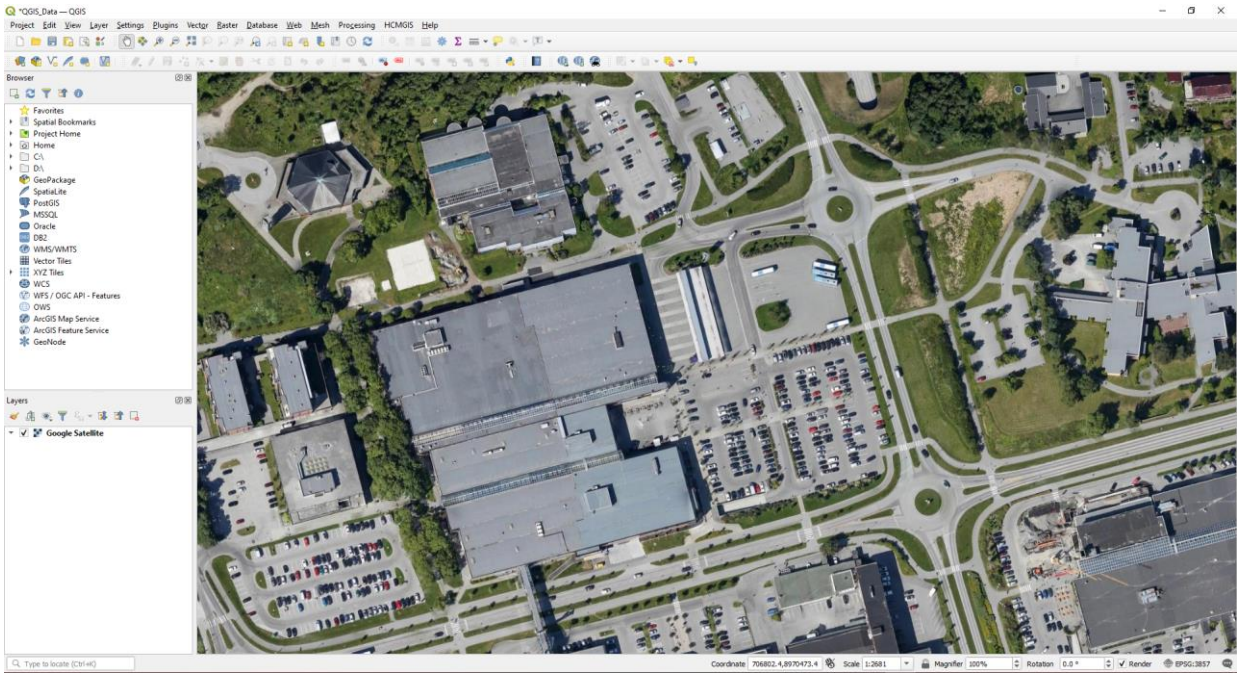


Figure 2-10. Selecting the study area in QGIS

❖ *Exporting Background Image*

From the main menu, select Project → Import/Export → Export Map to Image... (Figure 2-11).

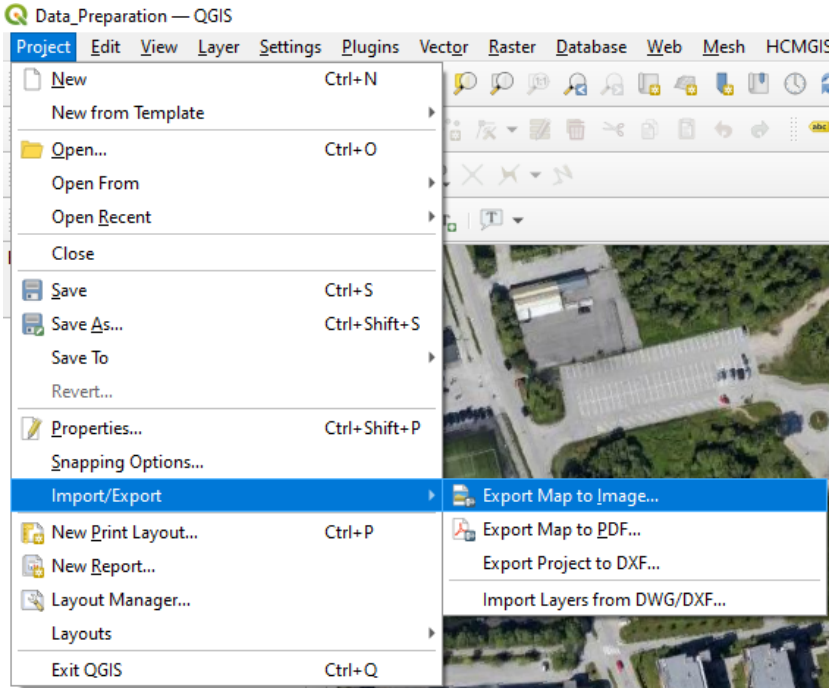


Figure 2-11. Exporting image in QGIS

In the next step, the user selects the boundary, resolution, and wanted information of the exported image. At this step, it is recommended to store 4-points locations and check the option ***"Append georeference information (embedded or via world file)"*** (Figure 2-12).

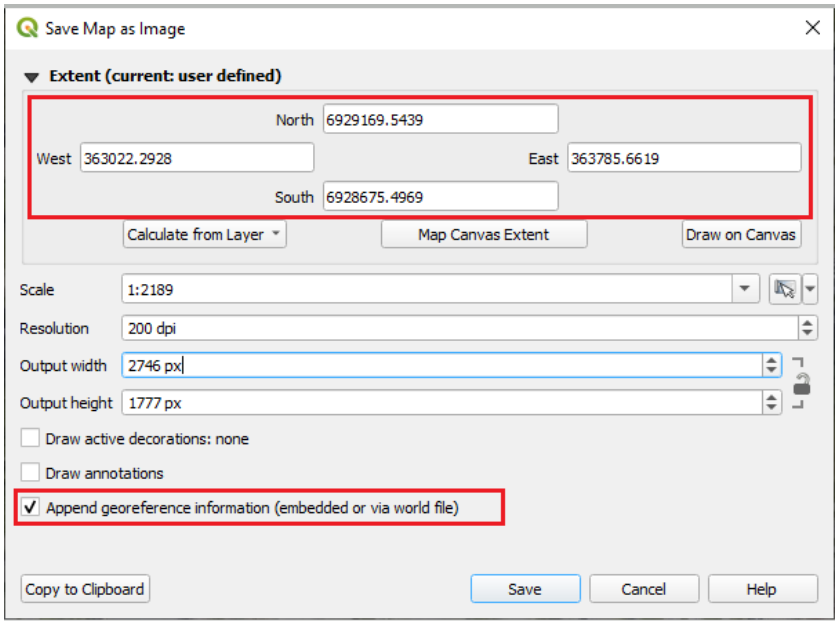


Figure 2-12. The exported image information

❖ *Assigning Background Image in SWMM*

In SWMM, the user adds a backdrop image by selecting: View → Backdrop → Load... The location of the background image and associated world coordinates file should be specified (Figure 2-13a). In the next step, the user specifies the coordinates of 4 corner points and the unit of the background image (Figure 2-13b). Finally, the backdrop image must be scaled to the map (Figure 2-13c).

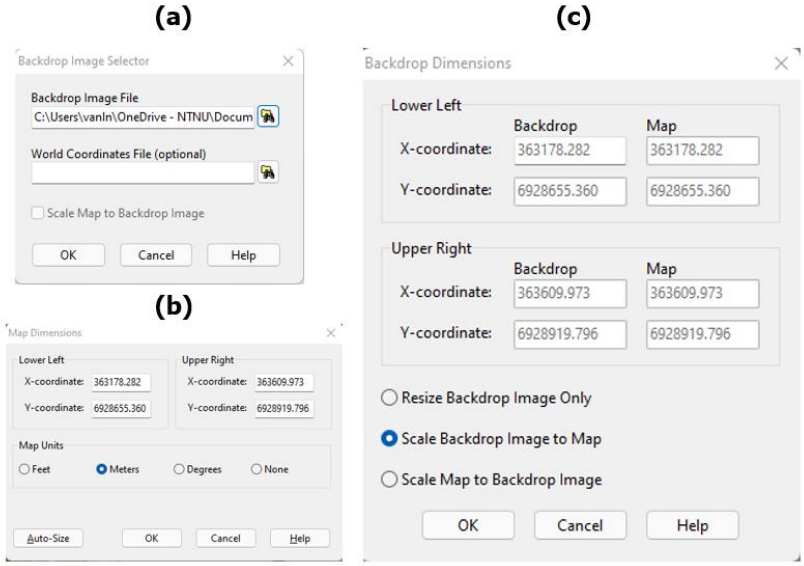


Figure 2-13. Loading a backdrop image into SWMM

After importing the background image into SWMM, the user should re-check the coordinates to make sure the image is in the correct coordinate system (Figure 2-14).

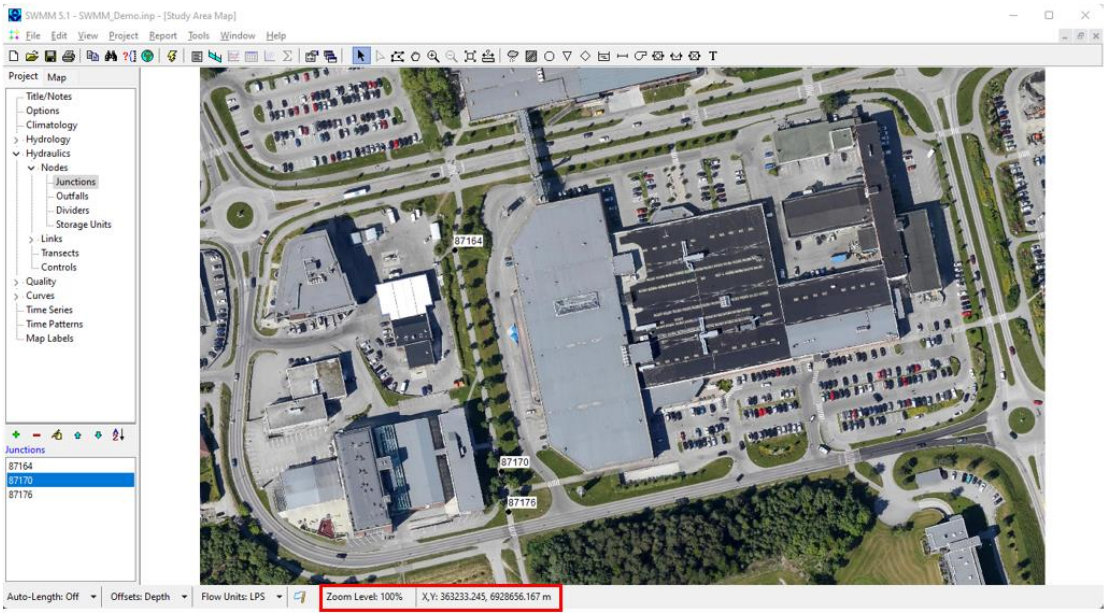


Figure 2-14. Background image in SWMM

b. Creating and Adjusting Attributes of Rain Gages

The process for creating a rain gage is illustrated in Figure 2-15.

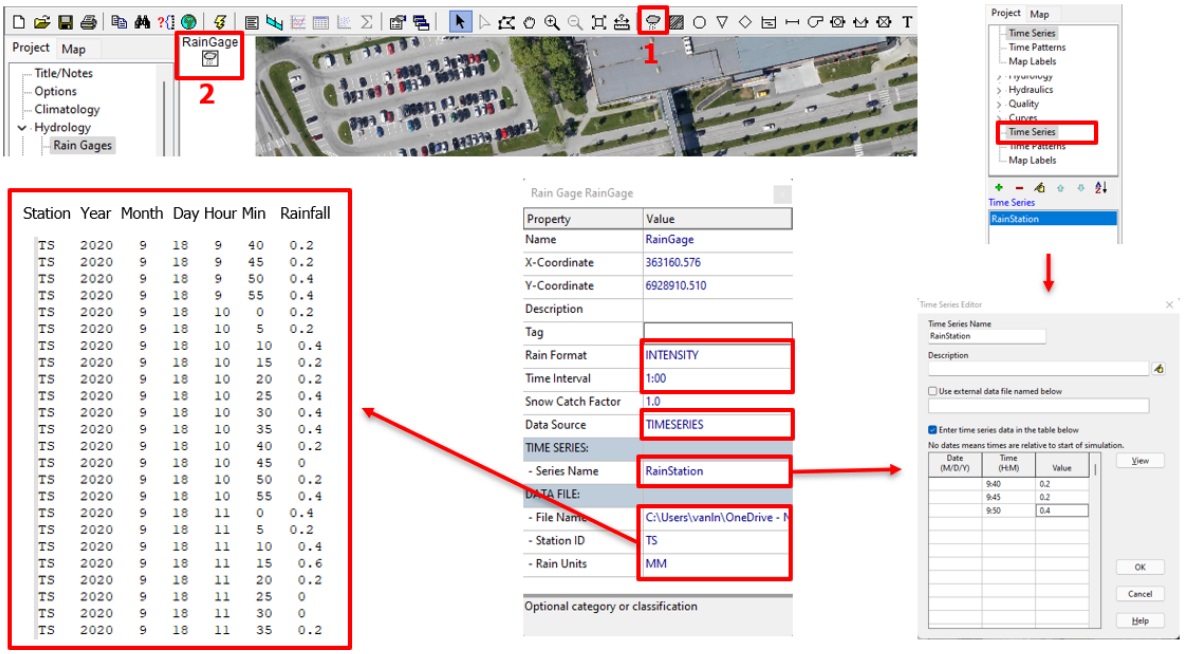


Figure 2-15. Creation of rain gage in SWMM

A rain gage gets rainfall data as an input source. The user can directly create time-series rainfall in SWMM or import rainfall data from outside (by selecting data sources as ***TIMESERIES*** or ***FILE***). The structure of the rainfall data file is represented in Figure 2-15. In the case of using rainfall data from an external file, the user needs to specify the file destination, name of the station, and unit of rainfall data (Figure 2-15).

c. Creating and Adjusting Attributes of Junctions

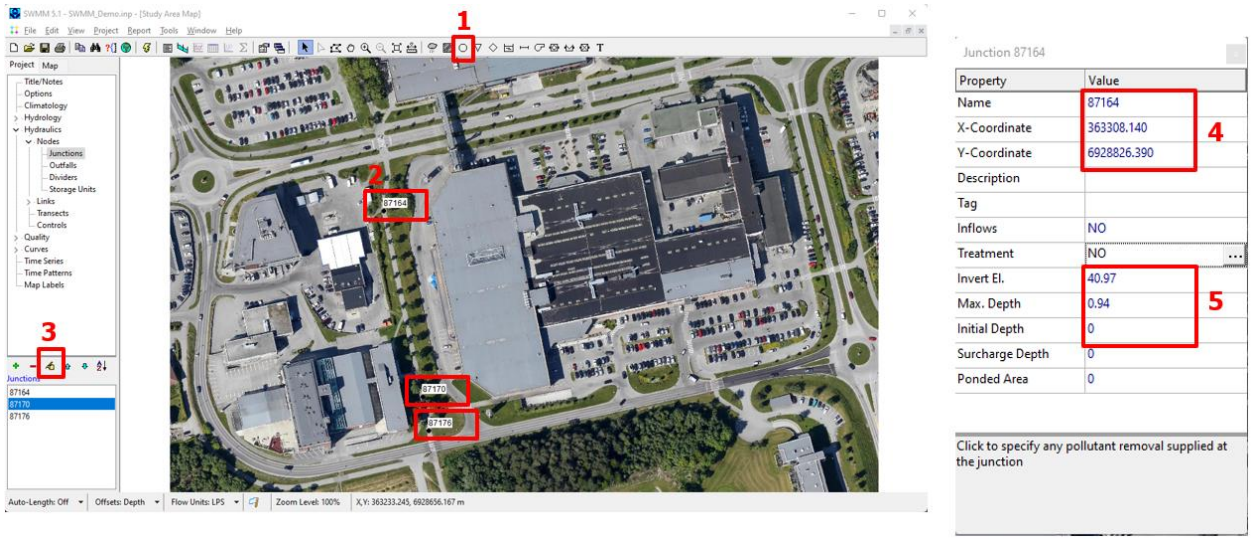


Figure 2-16. Junctions in SWMM

To create a junction in SWMM, the user must first select the junction symbol (Table 2-1). Next, the user points correct junction's location based on the background image. To adjust the coordinates of a particular junction, the user can double-click on the corresponding node and change the junction's parameters (step 3 in Figure 2-16).

Important parameters of a junction are name, coordinates, invert elevation (*Invert El.*), and maximum depth (*Max. Depth*) (steps 4 and 5 in Figure 2-16). Details of these parameters are illustrated in Figure 2-17.

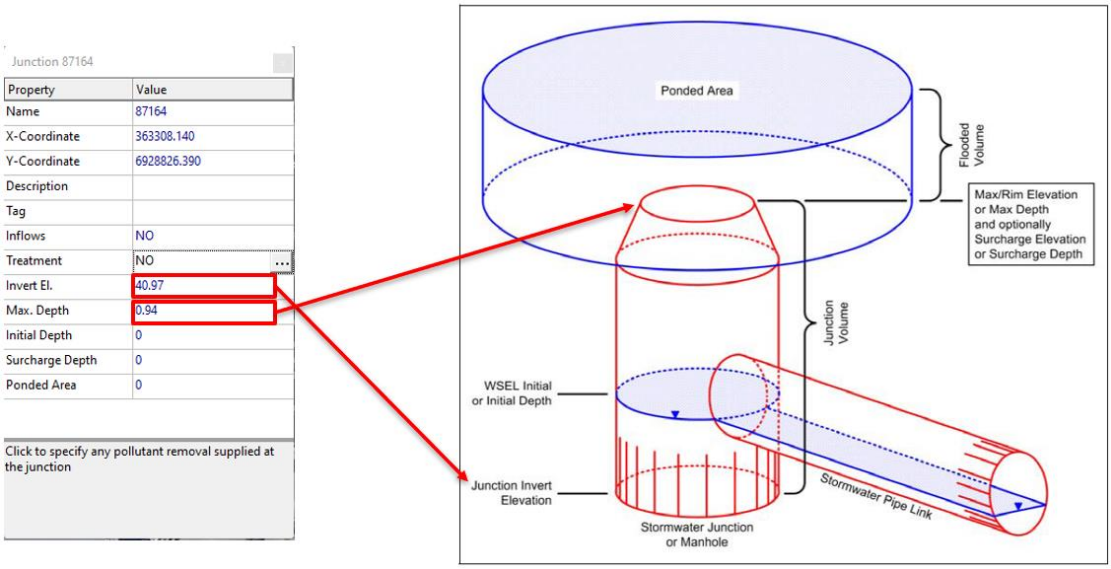


Figure 2-17. Some parameters of the junction

d. Creating and Adjusting Attributes of Conduits

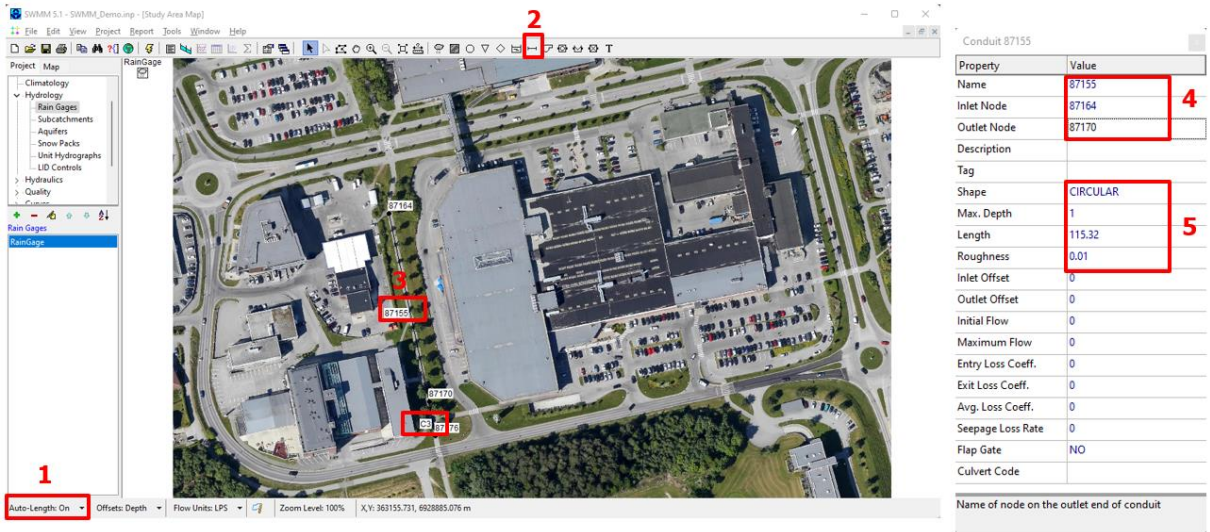


Figure 2-18. Conduits in SWMM

Before creating a conduit in SWMM, it is recommended to turn on the *Auto-Length* function

in SWMM (step 1 in Figure 2-18). This allows SWMM to calculate the length of each conduit automatically, otherwise, the user must type the length of a pipe after drawing. Some essential parameters of a conduit are name, inlet node, outlet node, shape, maximum depth (*Max. Depth*), length, and roughness (steps 4 and 5 in Figure 2-18).

Relating the shape and roughness values of the conduit, the user can refer to the values in Figure 2-19.

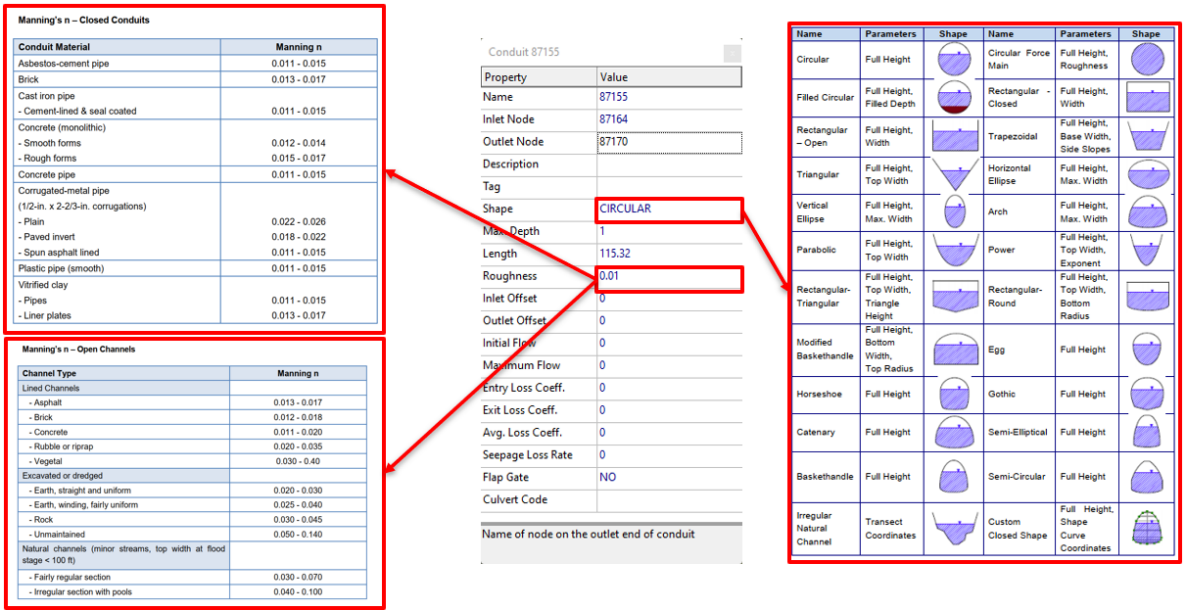


Figure 2-19. Some parameters of conduit

e. Creating and Adjusting Attributes of Sub-catchments

Steps for creating a sub-catchment in SWMM are represented in Figure 2-20.

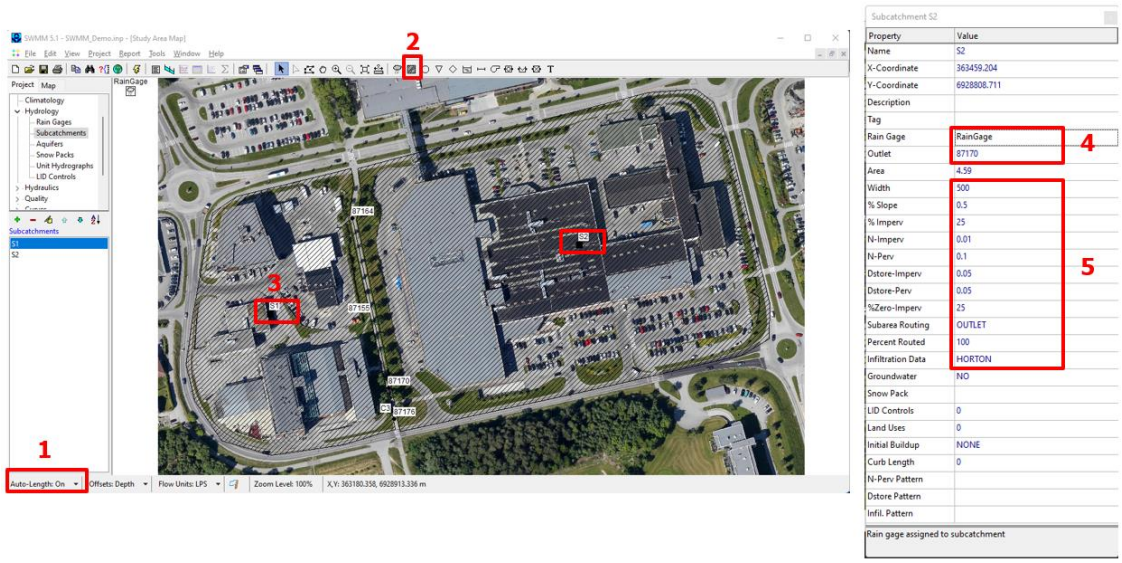


Figure 2-20. Sub-catchment in SWMM

Turning on the ***"Auto-Length"*** function allows SWMM to automatically calculate the area (in ha) of each sub-catchment while drawing. Delineating a sub-catchment highly depends on the user's purpose, study area, and the complexity of the topography. Therefore, in our opinion, the background image and the digital elevation model (DEM) are very important. For example, the determination of sub-catchment boundaries in urban areas is significantly different from the rural areas. Moreover, the transformation of water on complex surfaces is more difficult to observe than on simple/flat areas.

To create a sub-catchment, the user draws a polygon covering a specific area based on the study area, characteristics of the surface, and other elements (step 3 in Figure 2-20). Then, the outlet point of each sub-catchment must be specified depending on the slope of the surface in a particular sub-catchment (in general, the location of the outlet point is normally on the lowest point inside the sub-catchment because the water flows from the higher to lower location).

Essential parameters of each sub-catchment consist of rain gage, outlet point, area, width, the percentage of slope *(% Slope*), the percentage of imperviousness (*% Impev*), Manning's coefficient of impervious area (*N-Imperv*), Manning's coefficient of pervious area (*N-Perv*), depth of depression storage on the impervious/previous portion (*Dstore-Imperv/Dstore-Perv*) (steps 4 and 5 in Figure 2-20).

Details of sub-catchment parameters are shown in Figure 2-21.

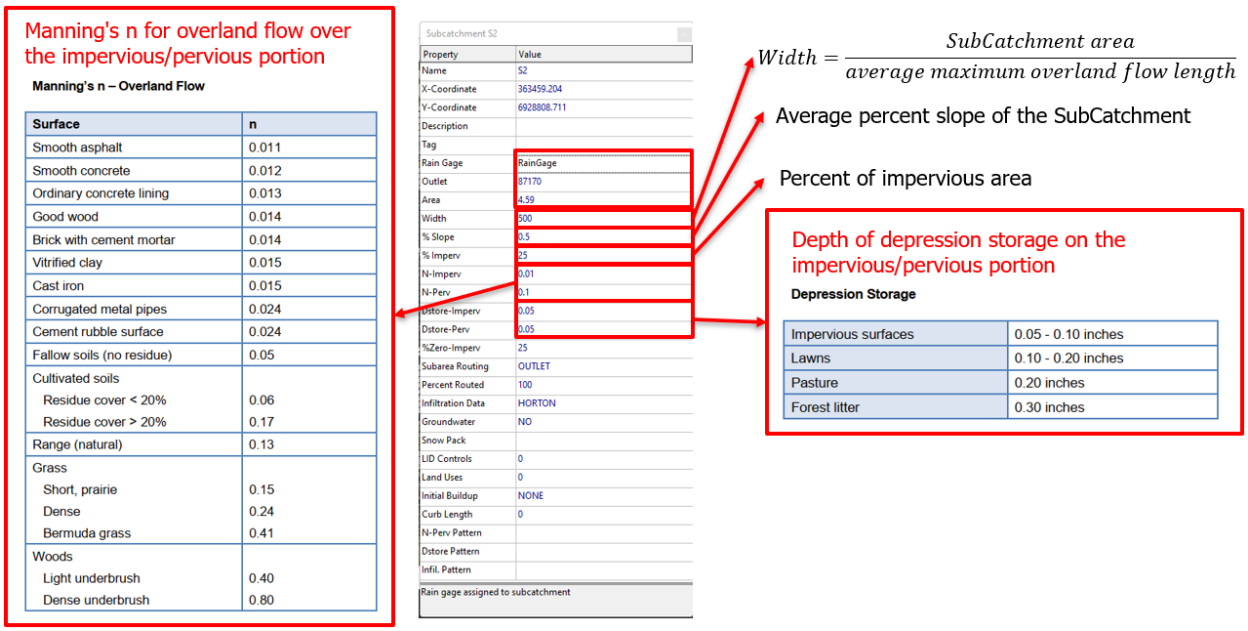


Figure 2-21. Parameters of sub-catchment

In general, the user must calculate the above parameters for each sub-catchment and import

them into SWMM. In this tutorial, we introduce to the user how to calculate the area, flow width, and the percentage of the slope of each sub-catchment using QGIS.

❖ *Computing Area of Sub-catchments*

Before calculating the area of sub-catchment in QSIG, the user needs to change the unit of area in QGIS to hectare (the default unit for area is square meters). To do this, in the main menu, select Project → Properties… On tab ***General*** in the ***Project Properties*** window, change the value in the option ***"Units for area measurement"*** from ***Square Meters*** to ***Hectares*** (Figure 2-22).

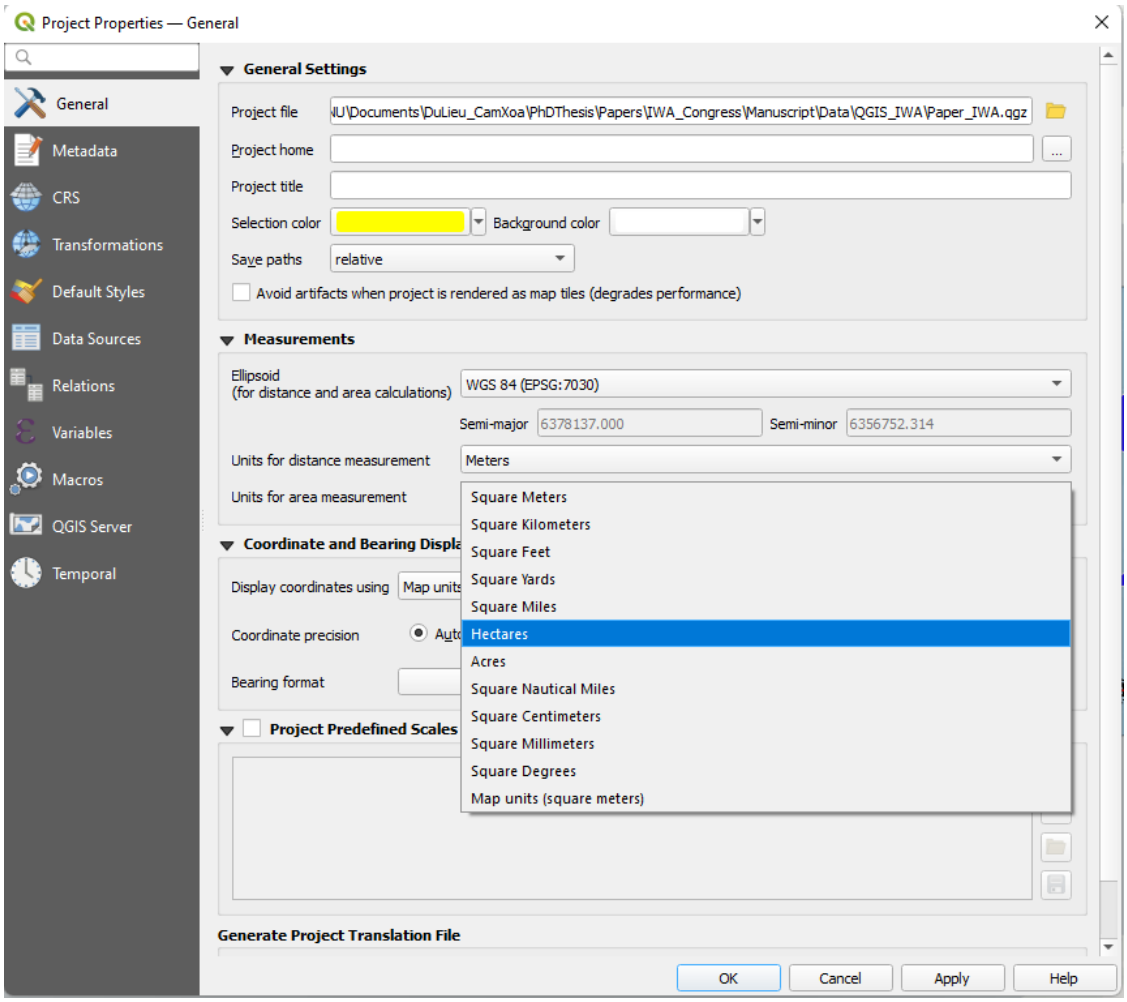


Figure 2-22. Changing the default unit in QGIS

After delineating the sub-catchment, the user creates a new column to calculate the area of sub-catchment in the attribute table. In this tutorial, we created a column called "Area" to store the area value of the sub-catchment. Steps for the calculating area are shown in Figure 2-23.

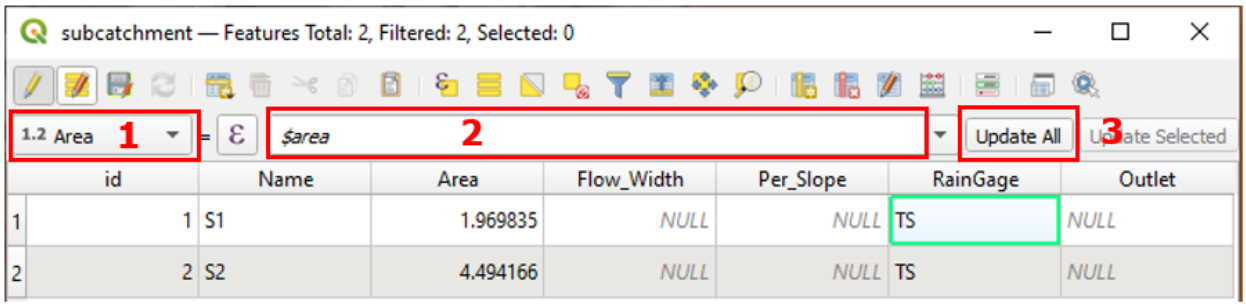


Figure 2-23. Calculation of area in QGIS

❖ *Computing Flow Width of Sub-catchments*

To compute flow width, DEM must be imported into QGIS. Then, the user run module ***"Flow Width and Specific Catchment Area"*** (Figure 2-24).

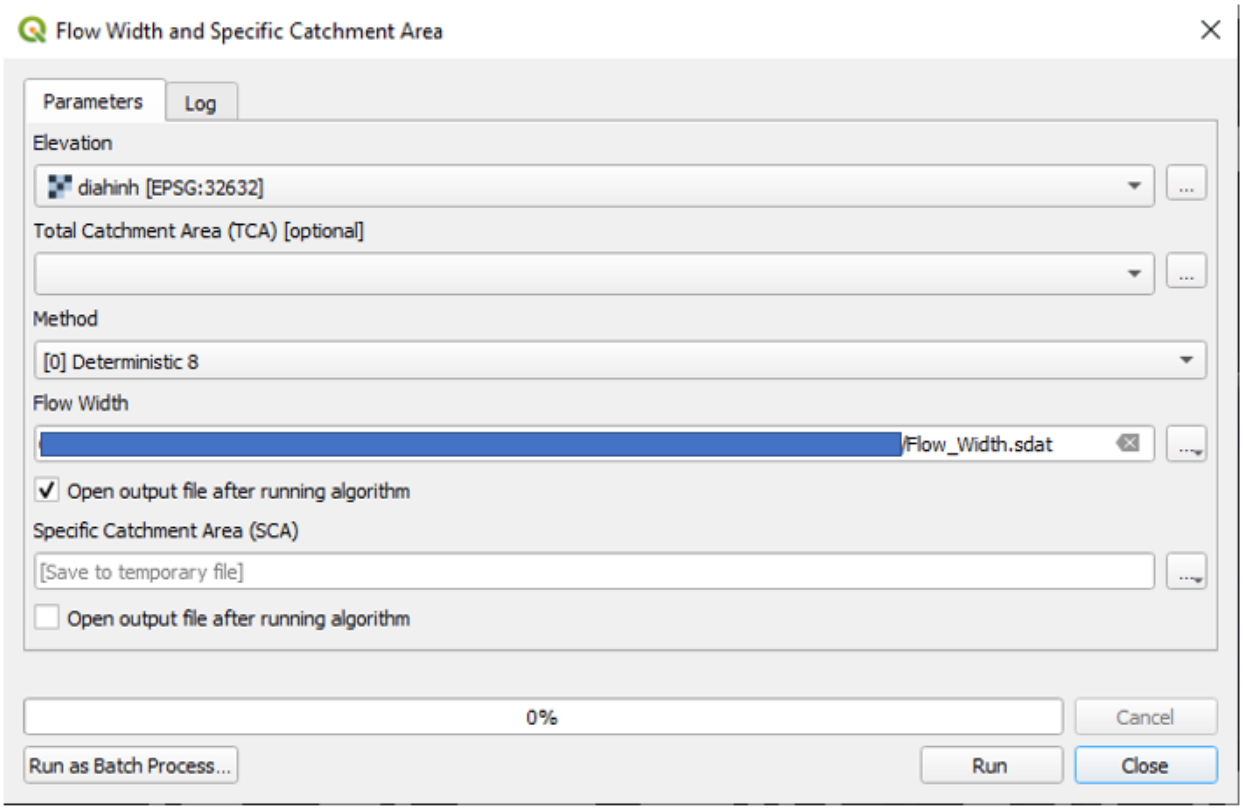


Figure 2-24. Computing flow width in sub-catchment

After having flow width values in the sub-catchment, the average value should be computed and assigned for a particular sub-catchment. To compute the mean flow width for each sub-catchment, the user needs to start running the ***"Zonal Statistics"*** function in QGIS (Figure 2-25).

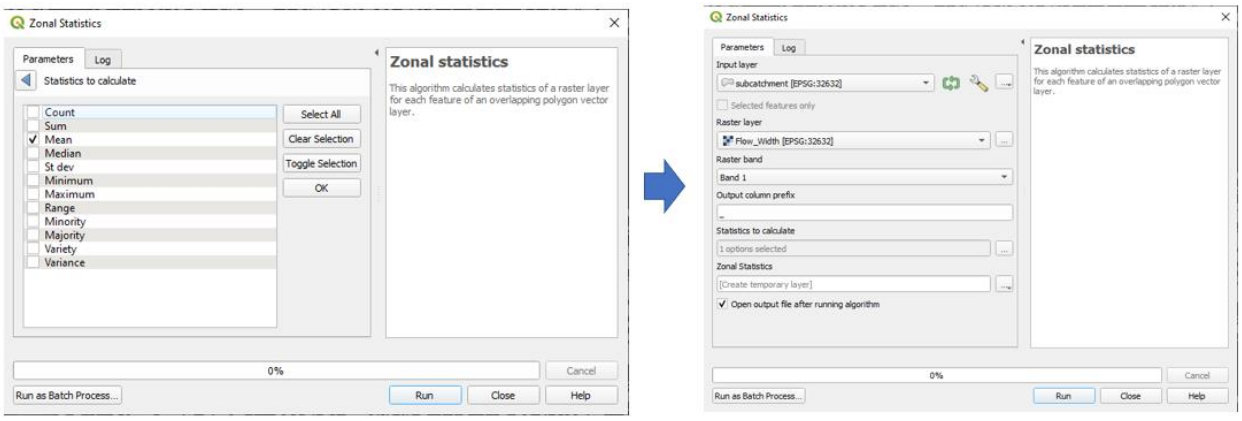
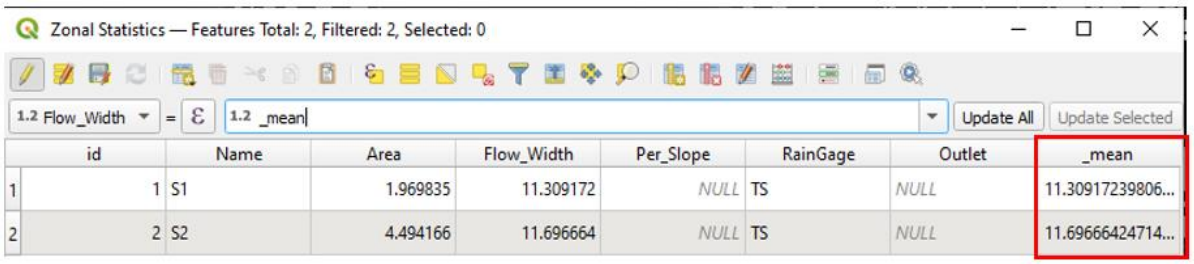


Figure 2-25. Calculating the mean flow width of sub-catchment in QGIS

The mean flow width value of each sub-catchment is shown in Figure 2-26.



	id	Name	Area	Flow_Width	Per_Slope	RainGage	Outlet	_mean
1	1	S1	1.969835	11.309172	NULL	TS	NULL	11.30917239806...
2	2	S2	4.494166	11.696664	NULL	TS	NULL	11.69666424714...

Figure 2-26. Mean flow width values of sub-catchment

❖ *Computing Percentage of Slope of Sub-catchments*

To compute slope in sub-catchment, the user needs to activate the ***"Slope"*** function in QGIS (Figure 2-27). The option ***"Slope expressed as percent instead of degrees"*** should be checked to compute the percentage of slope in the sub-catchment.

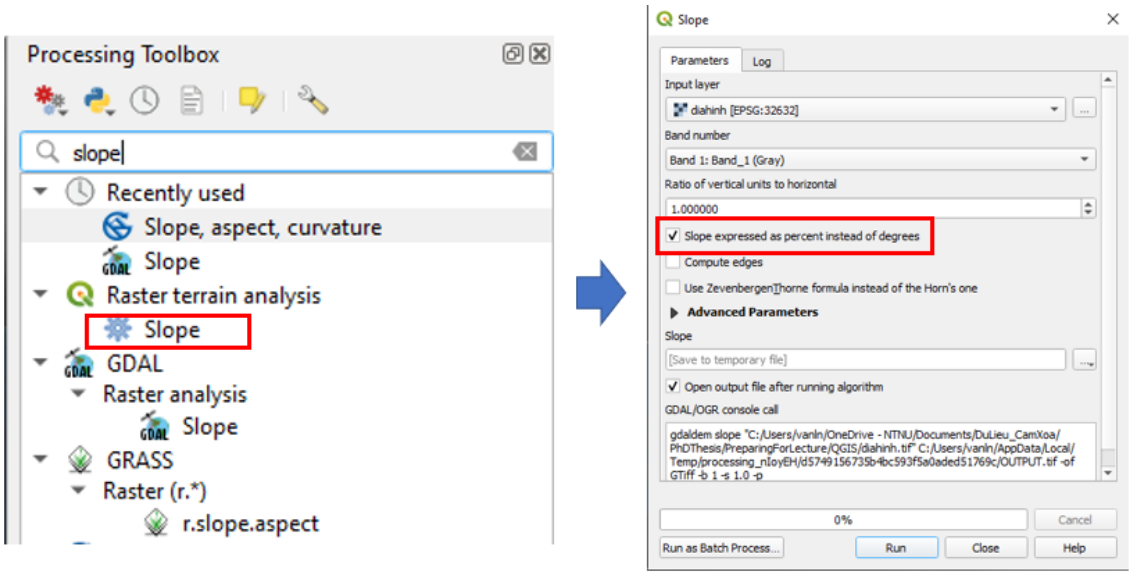
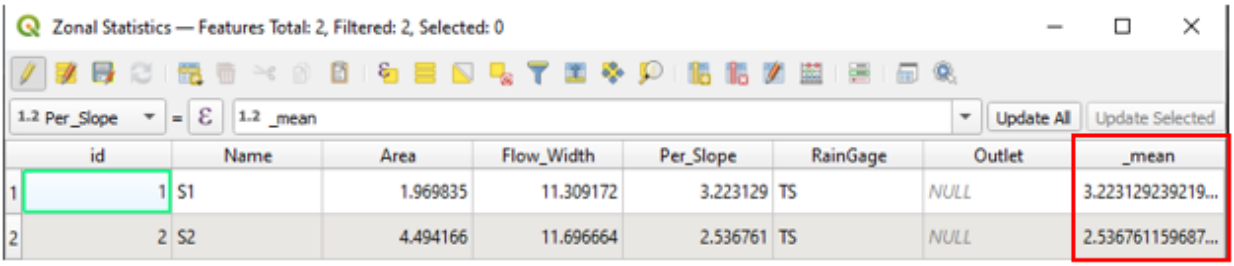


Figure 2-27. Computing slope of sub-catchment

The mean value of the percentage of slope should be computed for each sub-catchment (Figure 2-28).



	id	Name	Area	Flow_Width	Per_Slope	RainGage	Outlet	_mean
1	1	S1	1.969835	11.309172	3.223129	TS	NULL	3.223129239219...
2	2	S2	4.494166	11.696664	2.536761	TS	NULL	2.536761159687...

Figure 2-28. Mean of the percentage of slope

f. Creating and Adjusting Attributes Outfalls/Flow Dividers/Storage Units

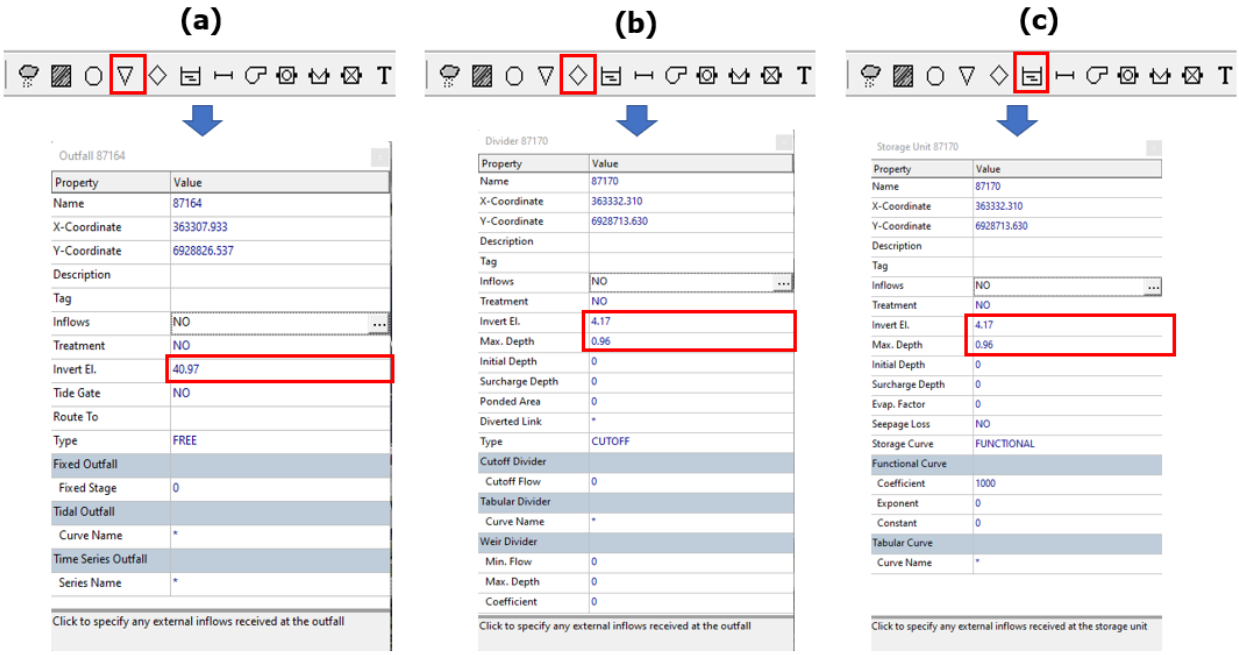


Figure 2-29. Creation of outfall (a), flow divider (b), and storage unit (c)

Figure 2-29 shows steps to create outfall (Figure 2-29a), flow divider (Figure 2-29b), and storage unit (Figure 2-29c) as well as their basic properties.

Creating and adjusting attributes of the outfall/flow divider/storage unit is as similar as creating and adjusting attributes of the junction. A simpler way to create the outfall/flow divider/storage unit is to convert from the junction (Figure 2-30).

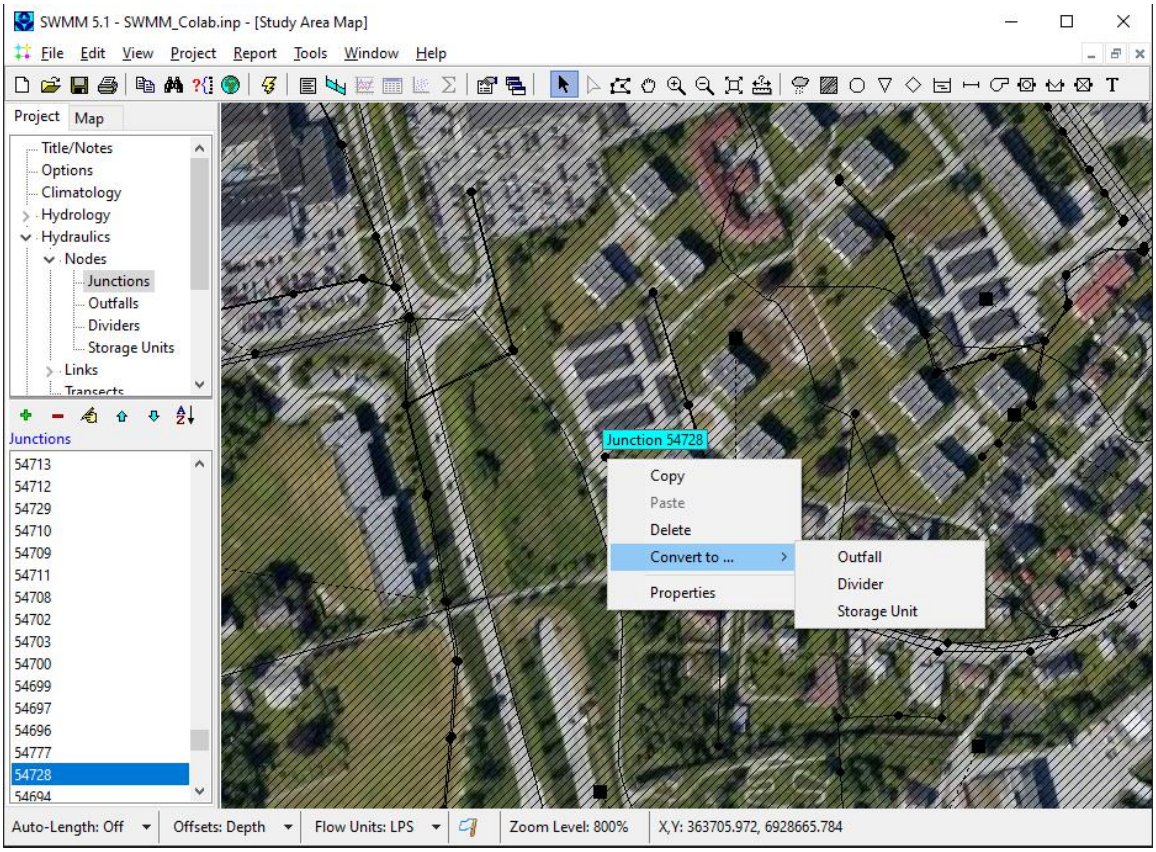


Figure 2-30. Converting a junction to an outfall, a divider, or a storage unit

g. Creating and Adjusting Attributes of Pumps/Orifices/Weirs/Outlets

Pumps, orifices, weirs, and outlets can be created and their attributes can be adjusted in the same way as the conduit. These components can be directly created in SWMM (Figure 2-31) or indirectly converted from conduits (Figure 2-32).

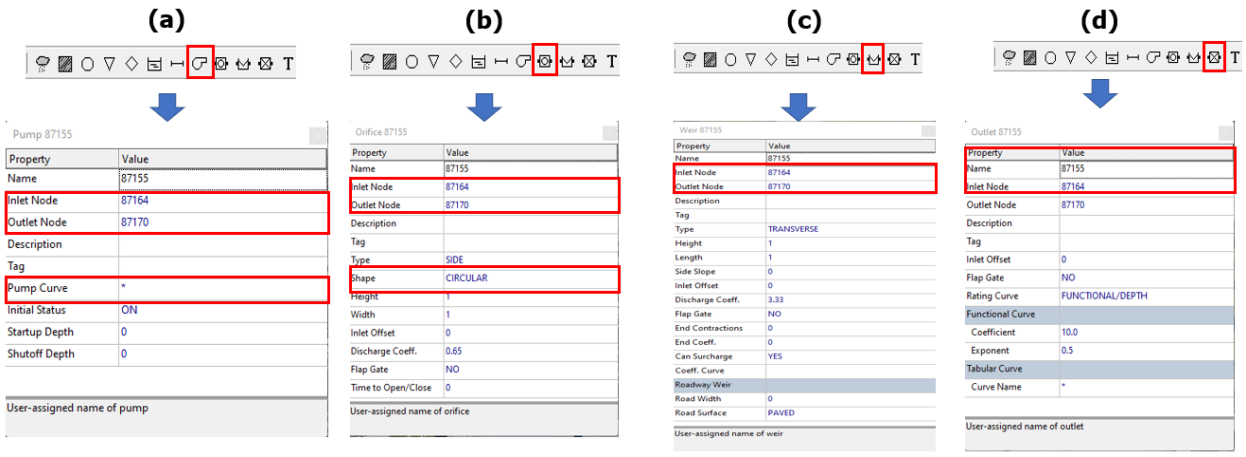


Figure 2-31. Creation of pump (a), orifice (b), weir (c), and outlet (d)

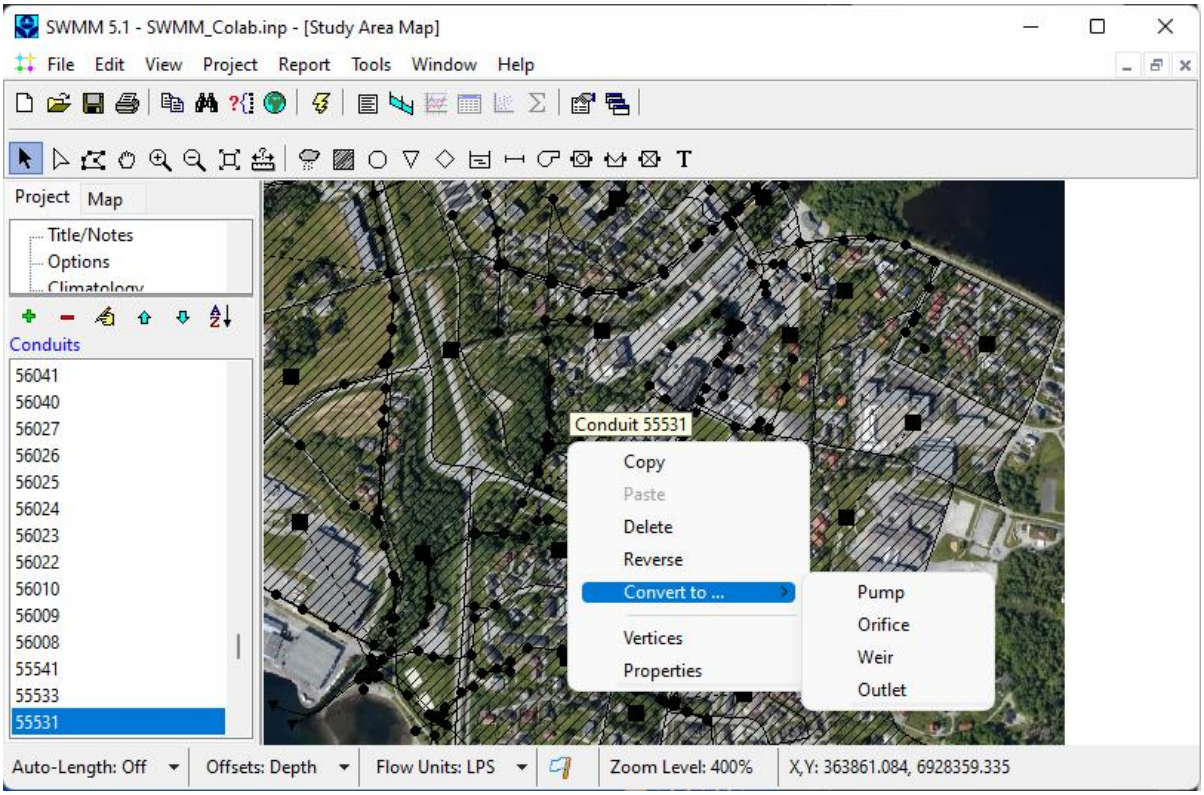


Figure 2-32. Converting a conduit to a pump, orifice, weir, or outlet

2.6. Data Transformation Tool

From the previous section, we can easily see that creating and adjusting attributes of SWMM components are time-consuming process and not easy for people who are not familiar with QGIS or geospatial software. Moreover, creating a sewer network manually is only suitable for a small area. Therefore, in this tutorial, we want to introduce a new transformation tool that can directly convert CSV files into an SWMM project.

A CSV (Comma-Separated Values) file is a text file that has a specific format that allows data to be saved in a table structured format. A CSV file is a simple text file that you can open in a wide variety of programs, including any program that works with plain text like the Notepad app.

Data for creating SWMM from CSV files and source code are available on GitHub: https://github.com/Lam-V-Nguyen/CSVToSWMM. Codes were written by python, the user therefore can open, modify, and run by using an integrated development environment (IDE) such as Spyder, PyCharm, Sublime Text, etc., In this tutorial, we used Spyder as the default IDE. After importing code into Spyder, the user can run code in Spyder directly by clicking on the ***"Run"*** button or hitting the ***F5*** key on the PC keyboard (Figure 2-33).

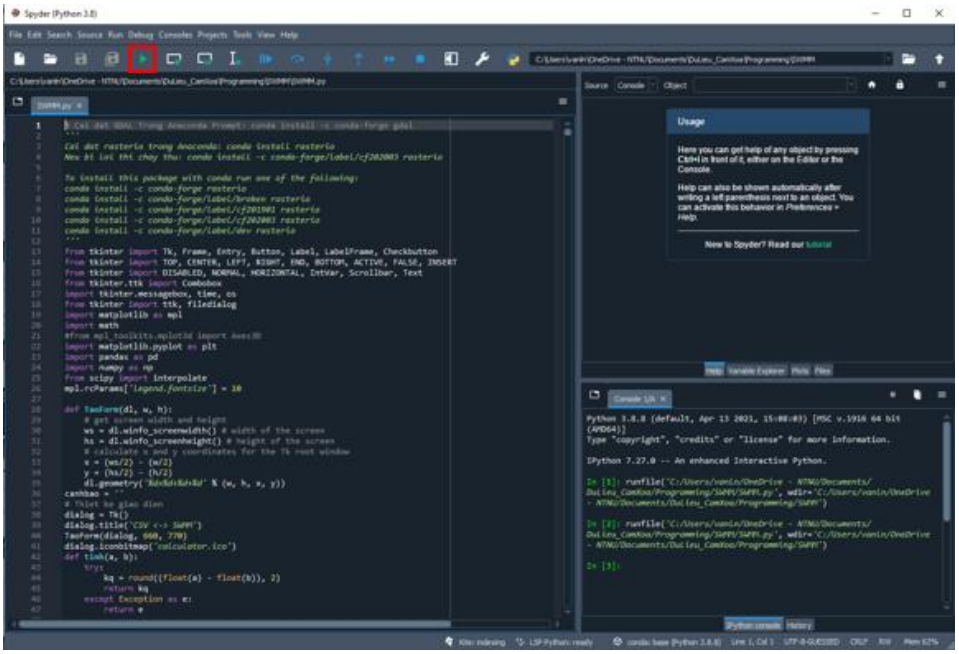
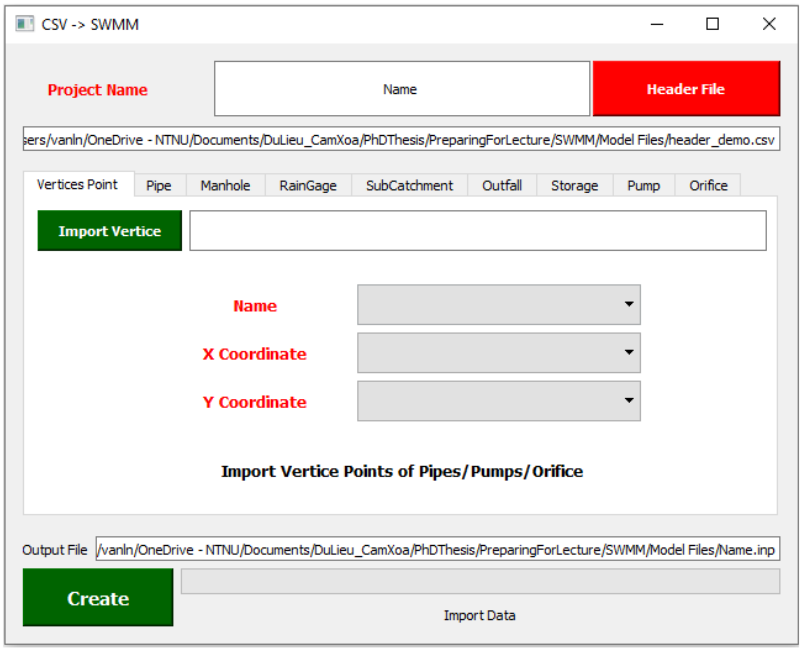


Figure 2-33. Spyder development environment

a. Importing default parameters for the SWMM project

The interface of this tool is shown in Figure 2-34a. The first step in creating an SWMM project using this tool is to set the name of the SWMM project. The default parameters of the SWMM project are declared in the header file. The structure of the header file is shown in Figure 2-34b.

(a) **(b)**



	A	B
1	Option	Value
2	FLOW_UNITS	LPS
3	INFILTRATION	MODIFIED_GREEN_AMPT
4	FLOW_ROUTING	DYNWAVE
5	LINK_OFFSETS	DEPTH
6	MIN_SLOPE	0
7	ALLOW_PONDING	NO
8	SKIP_STEADY_STATE	NO
9	START_DATE	9/21/2020
10	START_TIME	0:00:00
11	REPORT_START_DATE	9/21/2020
12	REPORT_START_TIME	0:00:00
13	END_DATE	9/22/2020
14	END_TIME	23:55:00
15	SWEEP_START	1-Jan
16	SWEEP_END	31-Dec
17	DRY_DAYS	0
18	REPORT_STEP	0:05:00
19	WET_STEP	0:05:00
20	DRY_STEP	1:00:00
21	ROUTING_STEP	0:00:30
22	RULE_STEP	0:00:00
23	INERTIAL_DAMPING	PARTIAL
24	NORMAL_FLOW_LIMITED	BOTH
25	FORCE_MAIN_EQUATION	H-W
26	VARIABLE_STEP	0.75
27	LENGTHENING_STEP	0
28	MIN_SURFAREA	0
29	MAX_TRIALS	0
30	HEAD_TOLERANCE	0
31	SYS_FLOW_TOL	5
32	LAT_FLOW_TOL	5
33	MINIMUM_STEP	0.5
34	THREADS	1

Figure 2-34. Data transformation tool interface

In the header file, the user only modifies the values in the second column. For details of these parameters, the user can refer to section ***"APPENDIX D – COMMAND LINE SWMM"*** in the manual provided in the references section of this tutorial. We do not recommend the user change this file before running the transformation tool. The user can change these parameters after importing the processed file into SWMM software.

b. Importing Vertices

The user must import vertices if any line components (conduits, pumps, orifices, weirs, or outlets) have more than 2 points. If this option is not implemented, all points on a line except the start and end points will be ignored (Figure 2-35a).

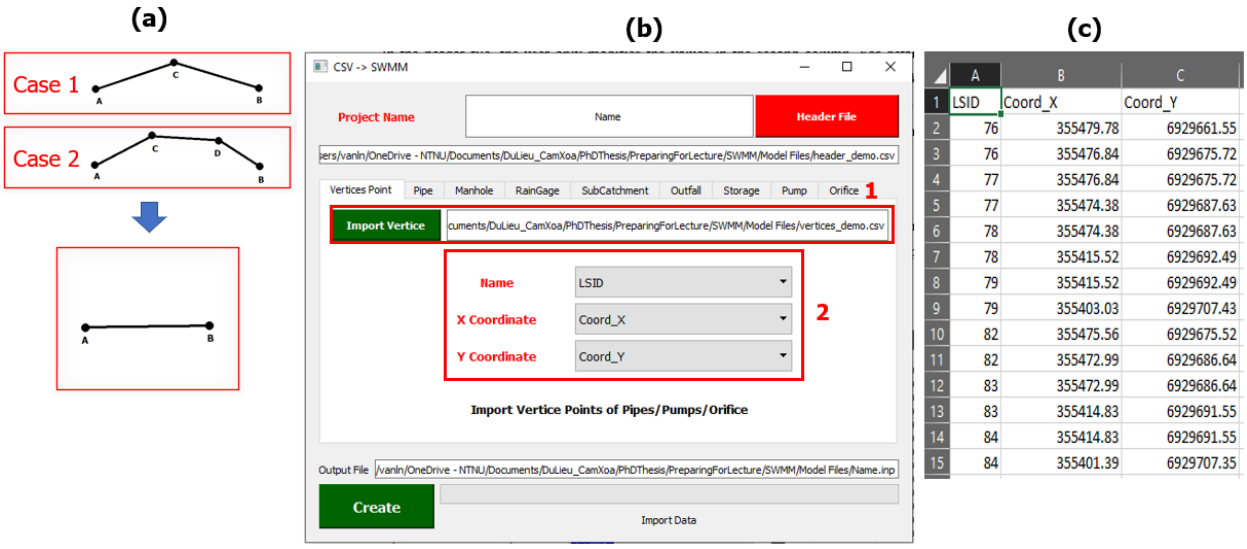


Figure 2-35. Converting vertices into SWMM

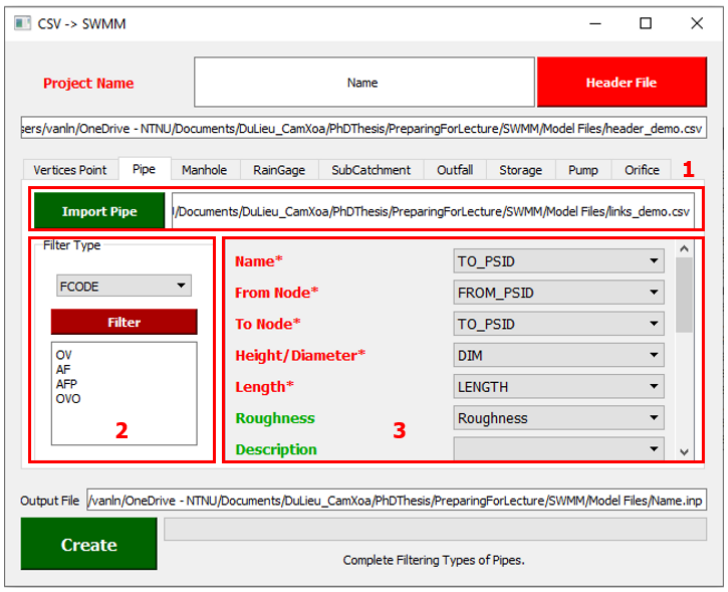
In the next step, the user must specify the location of the vertex file, and select the name and coordinate components of vertex points (steps 1 and 2 in Figure 2-35b). The structure of the vertex file is provided in Figure 2-35c.

c. Importing Conduits

Steps for importing conduits using this tool are represented in Figure 2-36a. Firstly, the user needs to specify the location of the conduit file (step 1 in Figure 2-36a). Secondly, the user can select which type of conduit to import into SWMM by choosing the particular column to keep. For example, we want to keep all conduits with ***FCODE*** corresponding to ***OV, AF, AFP***, and ***OVO*** (step 2 in Figure 2-36a). Finally, the user selects the column name corresponding to the listed attributes (step 3 in Figure 2-36a). In this step, the user can use the scrollbar to view more attributes. It is noted that the attributes in red are mandatory, and the attributes in green are

optional. If these attributes in green are not specified, the default values for these attributes will be used.

(a) **(b)**



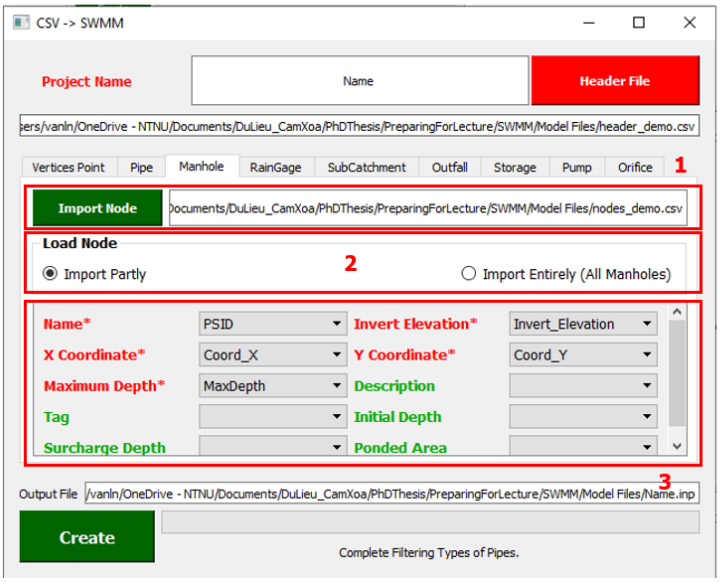
	A	B	C	D	E	F	G
1	LSID	FCODE	DIM	LENGTH	FROM_PSID	TO_PSID	Roughness
2	981	AF	0.23	46.93	151886	981_83019	0.020982467
3	1187	OV	0.16	25.06	459	54676	0.023396565
4	1221	SP	0.23	11.51	55137	1221_Outlet	0.024614898
5	1222	AF	0.3	66.61	472	54956	0.024749715
6	1229	SP	0.12	69.31	596	56495	0.02525517
7	1231	OV	0.15	2.85	1231_Inlet	153852	0.024614898
8	1272	SP	0.15	18.63	649	55950	0.02525517
9	1280	SP	0.15	16.14	653	56516	0.02525517
10	1328	OV	0.2	34.56	733	732	0.024405479
11	1330	OV	0.2	2.1	56353	152323	0.024405479
12	1341	AF	0.3	26.25	772	54663	0.022710703
13	1345	SP	0.16	15.98	775	89074	0.021309238
14	1346	SP	0.16	29.82	776	775	0.021309238
15	1358	OV	0.3	61.81	863	868	0.024405479
16	1363	SP	0.23	33.2	901	902	0.023549279
17	4956	OV	0.16	23.56	459	153455	0.023396565
18	4970	OV	0.2	23.83	1812	1813	0.021807907

Figure 2-36. Converting conduits into SWMM

The structure for the conduit file is shown in Figure 2-36b. The number of columns and the name of the column in this file depend on the number of required attributes and are totally defined by the user.

d. Importing Junctions

(a) **(b)**



	A	B	C	D	E
1	PSID	Coord_X	Coord_Y	Invert_Elevation	MaxDepth
2	981_83019	363749.23	6927915.28	6.0733	1.11
3	151886	363786.46	6927886.74	8.3218	1.54
4	54676	363131.55	6928394.67	25.8973	1.85
5	459	363136.16	6928370.04	24.5769	1.21
6	55137	364029.12	6928394.85	20.931	2.85
7	1221_Outlet	364018.29	6928398.76	21.3916	2.5
8	472	364142.57	6928717.76	23.3422	2.67
9	54956	364093.22	6928673.15	23.1971	3.07
10	596	364160.62	6929145.44	46.0245	1.46
11	56495	364161.72	6929076.14	43.4393	1.27
12	1231_Inlet	364097.85	6928460.76	21.5958	3.27
13	153852	364095.62	6928459	21.4892	3.38
14	649	363933.52	6928924.23	35.4189	1.21
15	55950	363914.96	6928922.67	34.368	1.9

Figure 2-37. Converting junctions into SWMM

To import junctions into SWMM via this tool, the user must define the location of the junction file (step 1 in Figure 2-37a). Next, the user can select import entire junctions or only junctions that associate with conduits, pumps, or orifices (step 2 in Figure 2-37a). In the final step, the user selects the column name corresponding to the listed attributes (step 3 in Figure 2-37a).

The structure for the conduit file is shown in Figure 2-37b. The number of columns and the name of the column in this file depend on the number of required attributes and are totally defined by the user.

e. Importing Rain Gages

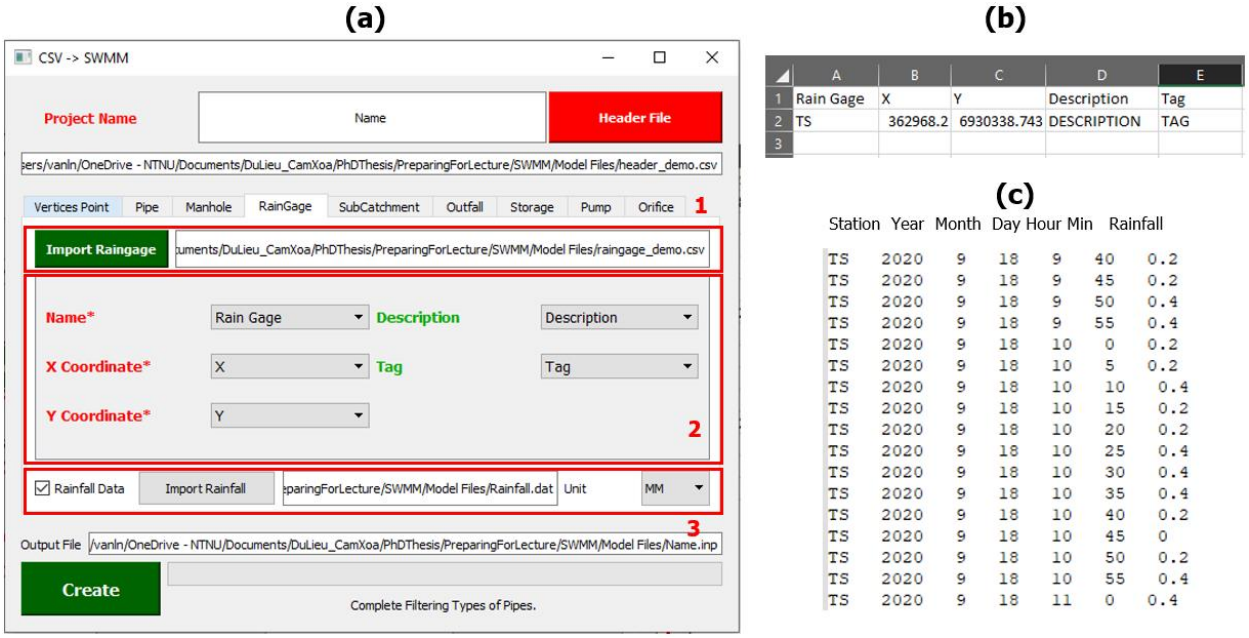


Figure 2-38. Converting rain gage into SWMM

Before importing rain gage into SWMM, the user must specify the location of the rain gage and assigns attributes of rain gage according to listed attributes (steps 1 and 2 in Figure 2-38a). After that, if the user wants to use external rainfall data, the location and unit of rainfall data have to be specified (step 3 in Figure 2-38a). The structure of rain gage and rainfall data are illustrated in Figure 2-38b and Figure 2-38c, respectively.

f. Importing Sub-catchments

After specifying the location of the sub-catchment file, the user must assign the attributes of the sub-catchments (Figure 2-39). Some extra values such as area, flow width, and the percentage of slope can be added into new columns in the CSV file and imported into SWMM via this tool.

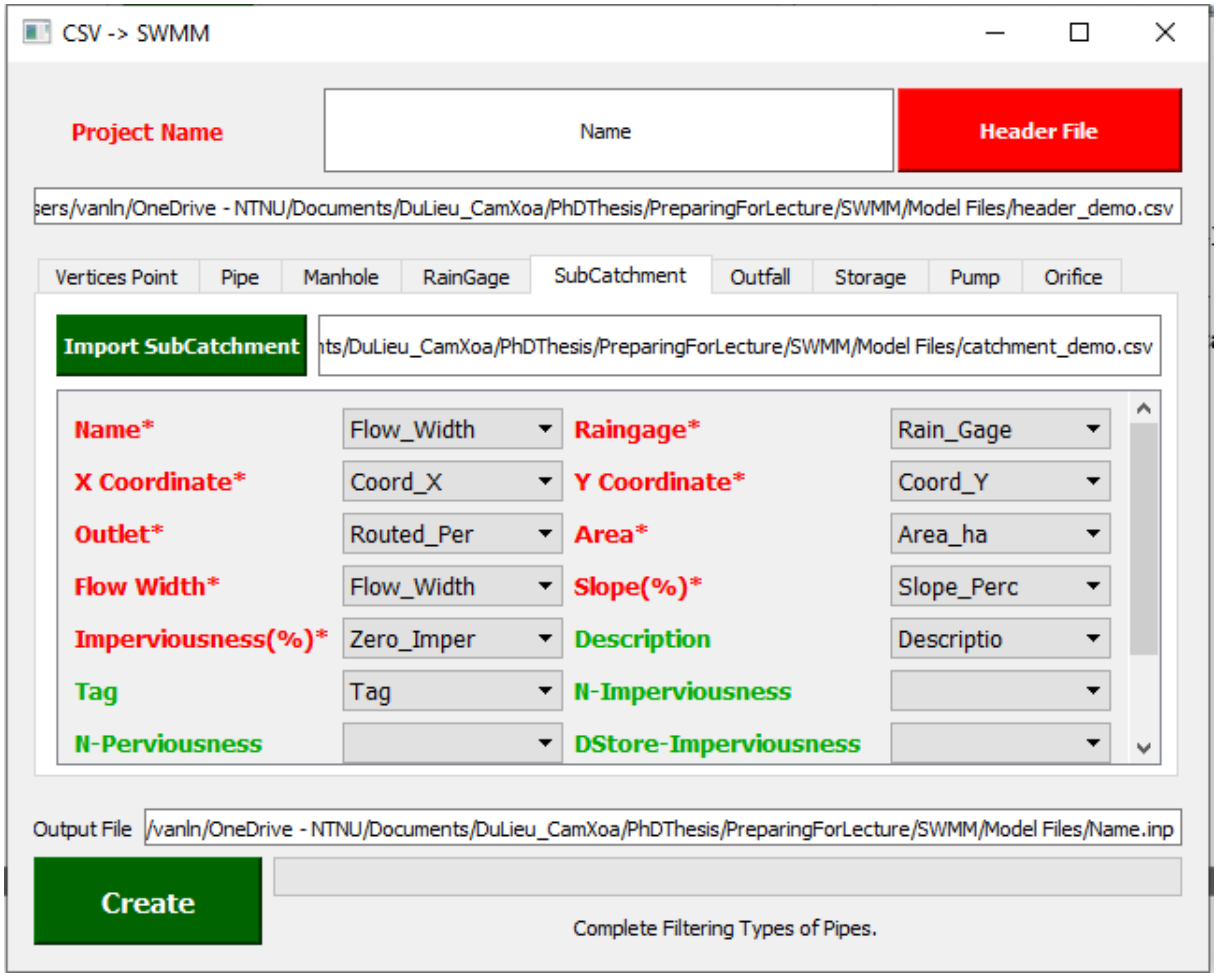


Figure 2-39. Converting sub-catchments into SWMM

The attributes of sub-catchments are constructed in Figure 2-40. It is worth noticing that all attributes of a sub-catchment (except coordinate elements) are unchanged.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	Name	Coord_X	Coord_Y	Area_ha	Flow_Width	Descriptic	Tag	Rain_Gage	Outlet	Slope_Per	Imperv_pe	N_Imperv	N_Perv	Dstore_Pe	Dstore_Im	Zero_Impe	Routed_Per
2	SubCatch_128	363705.483	6927924.092	2.8	975.79			TS	54190								
3	SubCatch_128	363710.708	6927995.432	2.8	975.79			TS	54190								
4	SubCatch_128	363705.224	6927991.432	2.8	975.79			TS	54190								
5	SubCatch_128	363684.224	6927978.432	2.8	975.79			TS	54190								
6	SubCatch_128	363673.224	6927969.278	2.8	975.79			TS	54190								
7	SubCatch_128	363674.881	6927967.27	2.8	975.79			TS	54190								
8	SubCatch_128	363683.039	6927946.538	2.8	975.79			TS	54190								
9	SubCatch_128	363681.333	6927923.521	2.8	975.79			TS	54190								
10	SubCatch_128	363671.846	6927871.158	2.8	975.79			TS	54190								
11	SubCatch_128	363651.271	6927832.965	2.8	975.79			TS	54190								
12	SubCatch_128	363632.656	6927813.742	2.8	975.79			TS	54190								
13	SubCatch_128	363597.936	6927793.248	2.8	975.79			TS	54190								
14	SubCatch_128	363569.66	6927787.347	2.8	975.79			TS	54190								
15	SubCatch_128	363551.937	6927789.054	2.8	975.79			TS	54190								

Figure 2-40. The structure data of sub-catchment

g. Importing Outfalls

Steps 1 and 2 in Figure 2-41a show how to import outfall into SWMM using this tool. Figure 2-41b describes the structure of outfall data using this transformation tool.

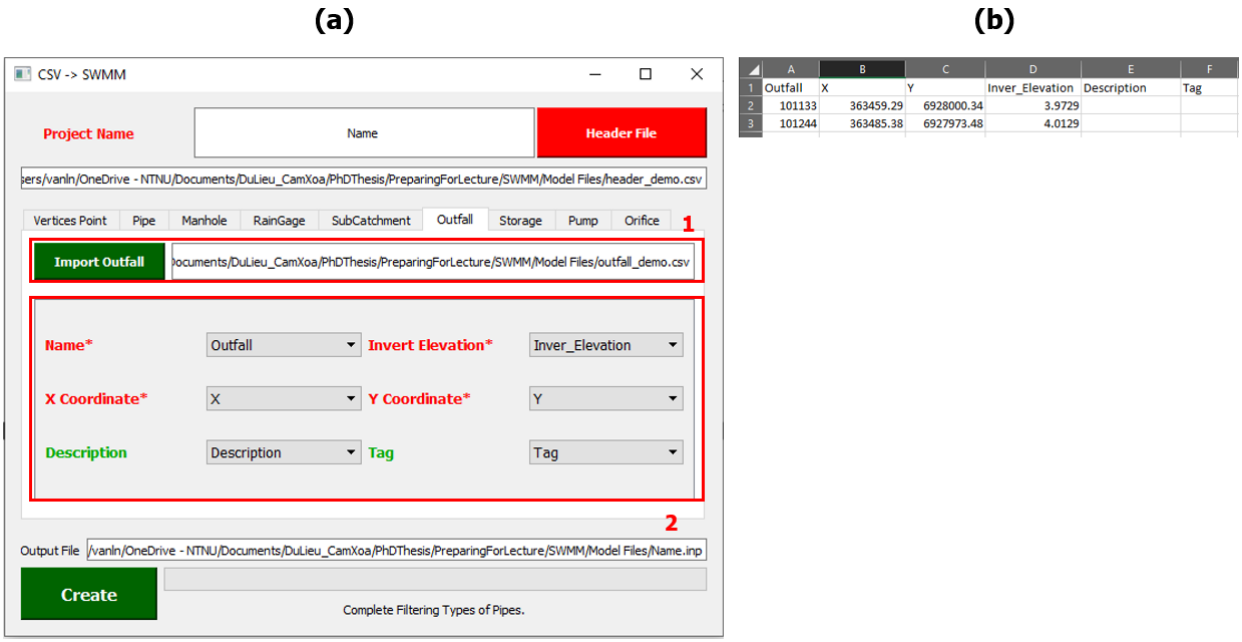


Figure 2-41. Converting outfalls into SWMM

h. Importing Storage Units, Pumps, and Orifices

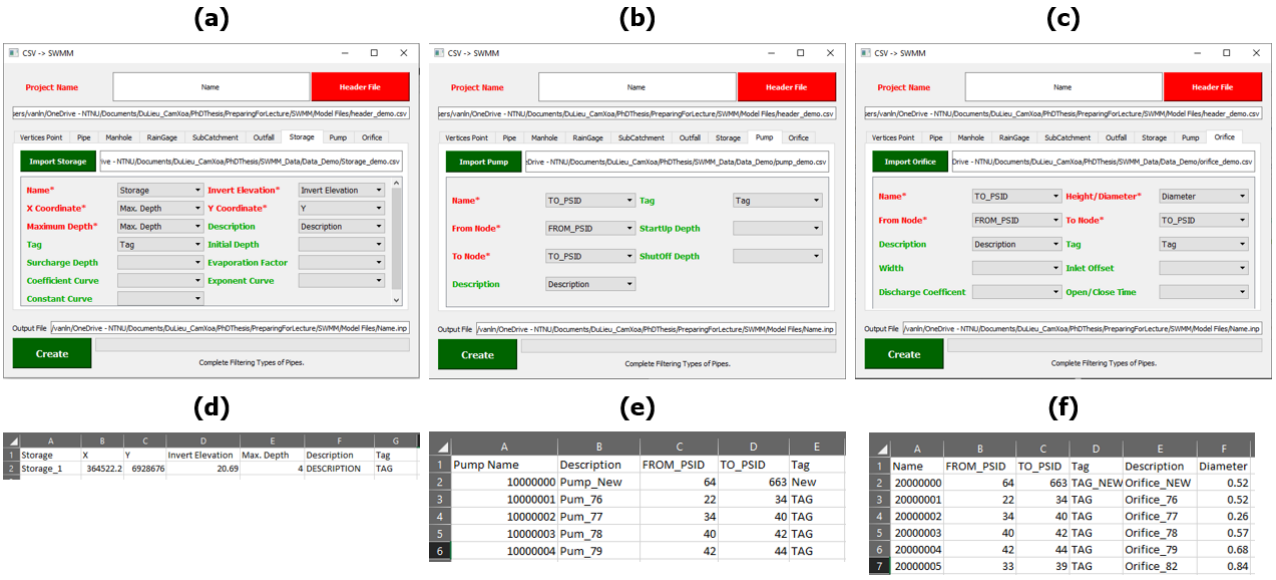


Figure 2-42. Converting storage units, pumps, and orifices into SWMM

Importing storage units, pumps, and orifices is similar to importing outfall into SWMM (Figure 2-42). The structure data of the storage unit, pump, and orifice files are represented in Figure 2-42a, Figure 2-42b, and Figure 2-42c respectively.

After all of the components have been imported into the transformation tool, the transformation process will be implemented after the user clicks on the ***"Create"*** button (Figure 2-43). The progress bar shows the complete percentage of the current transformation.

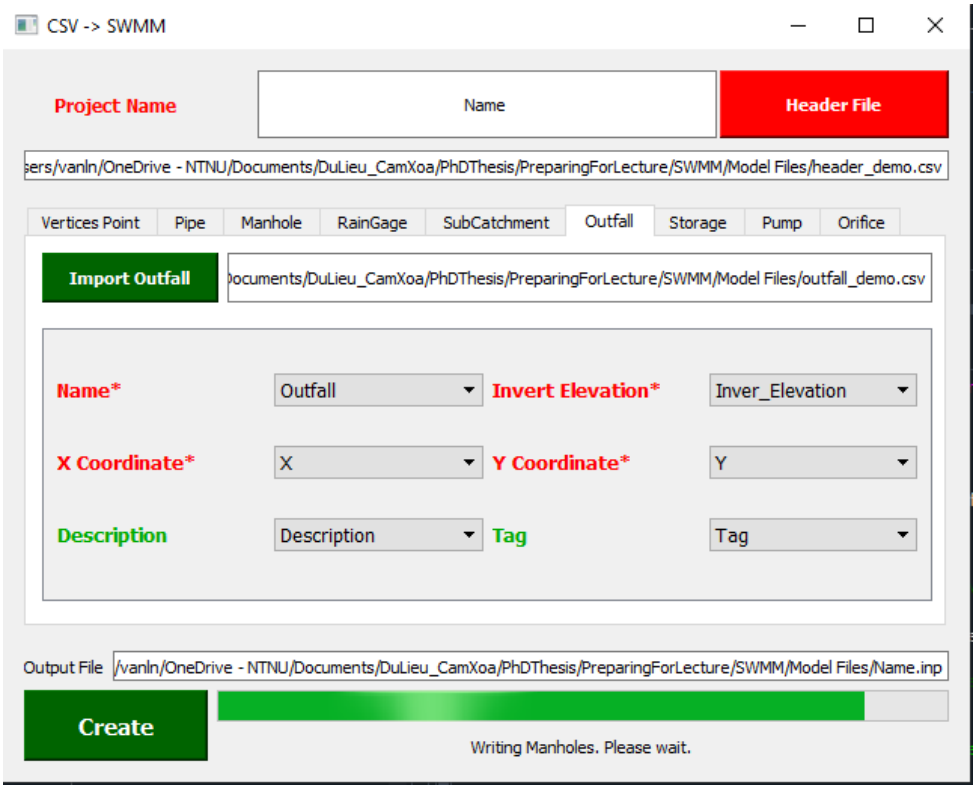


Figure 2-43. Importing data into SWMM

After the transformation is completed, an INP file will be created. The user can start SWMM software and import this file to check the network (Figure 2-44).

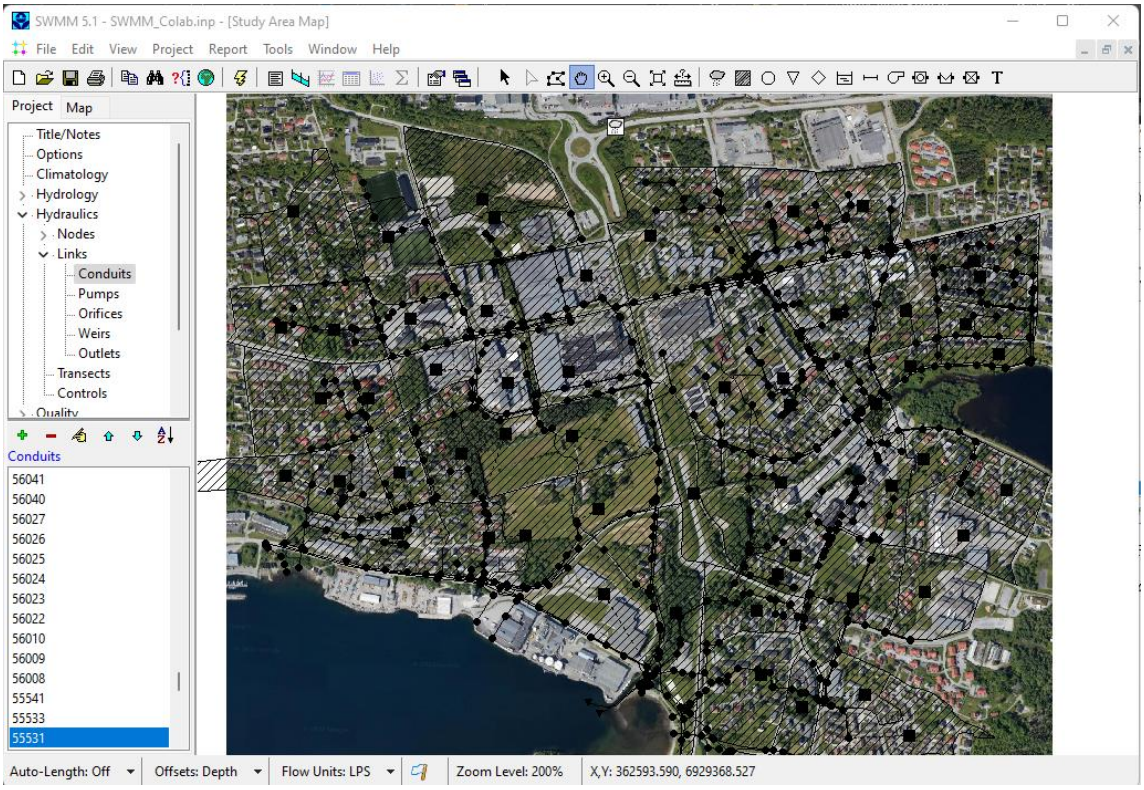


Figure 2-44. The sewer network in SWMM

3. Starting a Simulation and Viewing Results

3.1. Starting a Simulation

After checking the network in SWMM and ensuring the values of components are consistent, the user can run a simulation.

To simulate a network SWMM, the user selects ***Project → Run Simulation*** on the main menu (Figure 3-1).

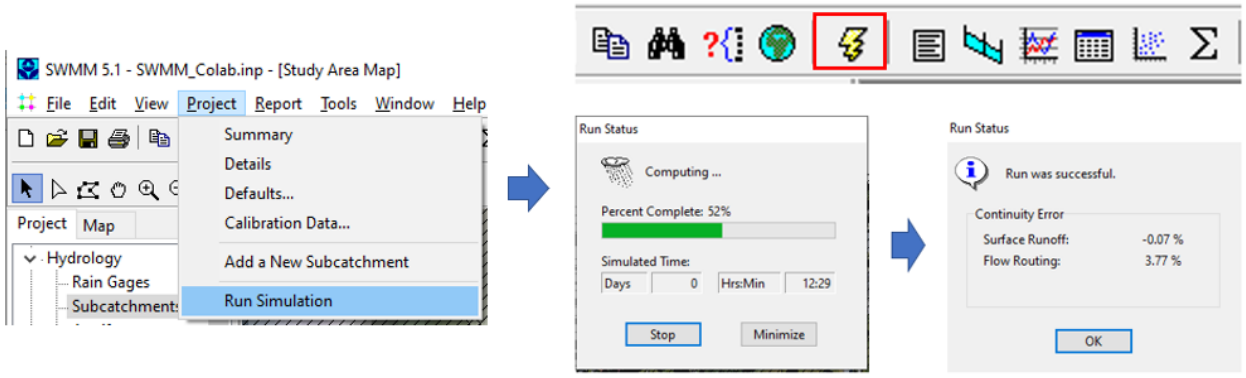


Figure 3-1. Starting a simulation in SWMM

After the simulation runs successfully, the user can see the time bar on the top right corner of the interface SWMM software (Figure 3-2).

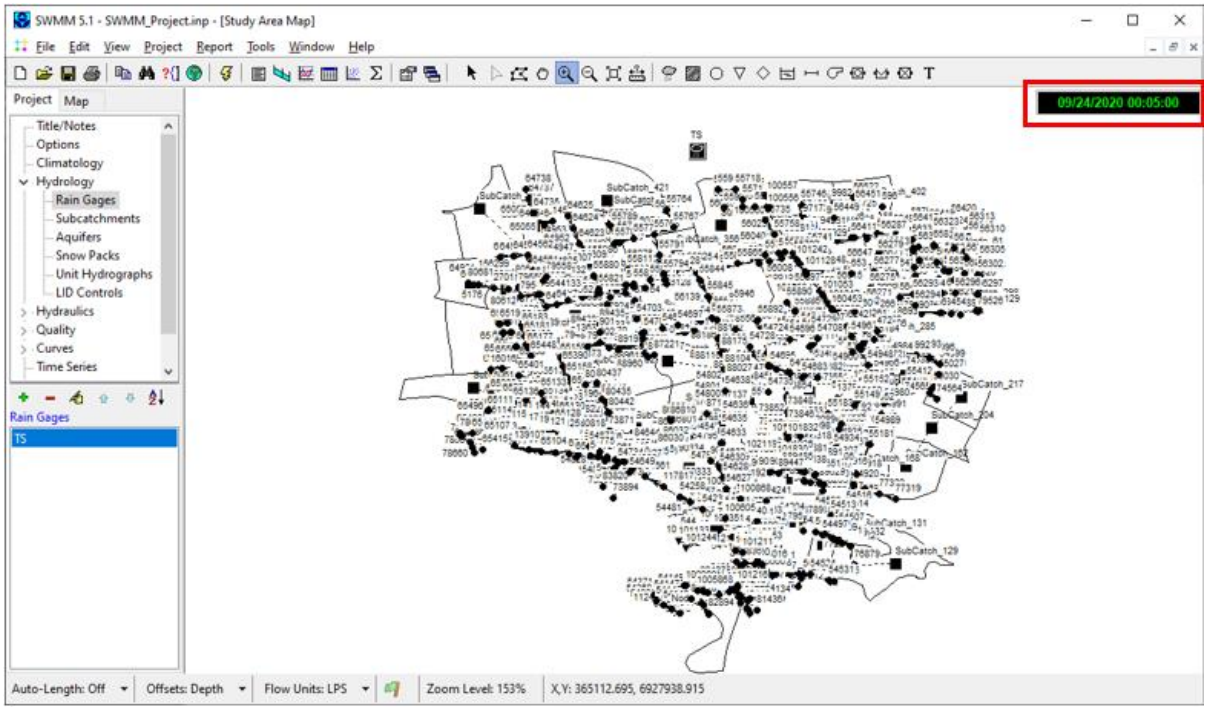


Figure 3-2. A successful simulation

SWMM allows the user to see dynamic visualization with a successful simulation. To see a

dynamic simulation, the user opens the tab “Map” in the SWMM interface (Figure 3-3).

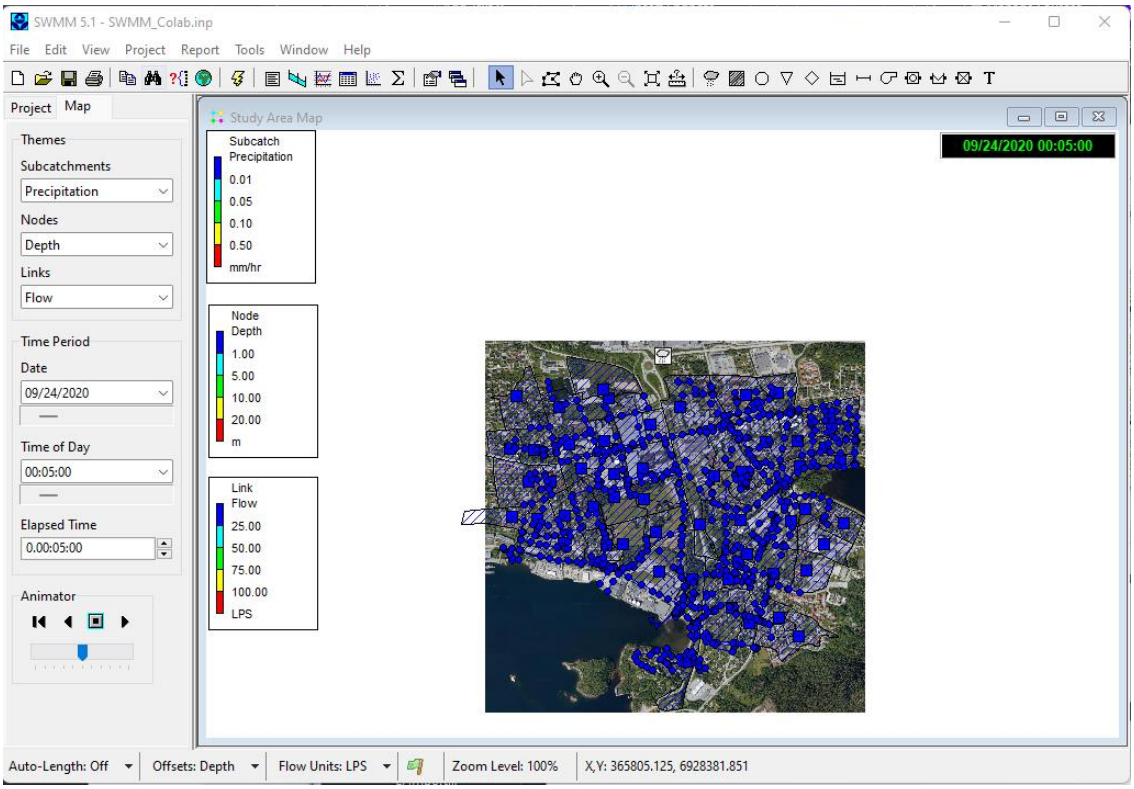


Figure 3-3. Dynamic visualization window in SWMM

In this window, the user can dynamically visualize some characteristics of the sub-catchment (Figure 3-4a), the junction (Figure 3-4b), and the conduit (Figure 3-4c).

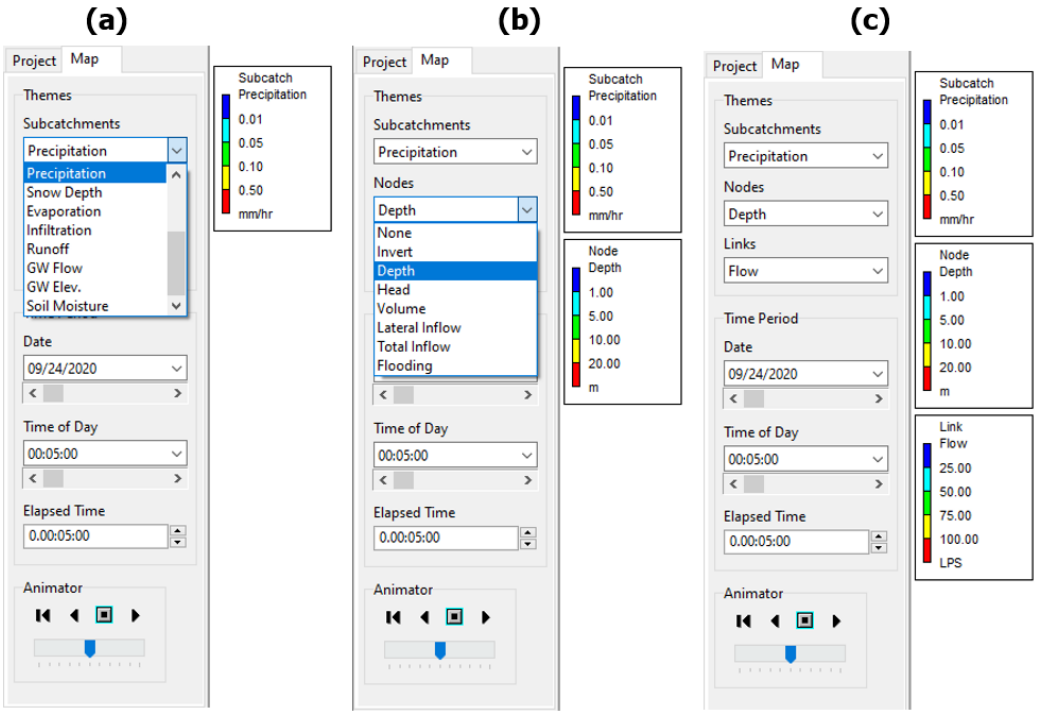


Figure 3-4. Component's properties for dynamic visualization in SWMM

The user can see many properties of the components during the simulation period by choosing functions in the toolbar menu (Figure 3-5).

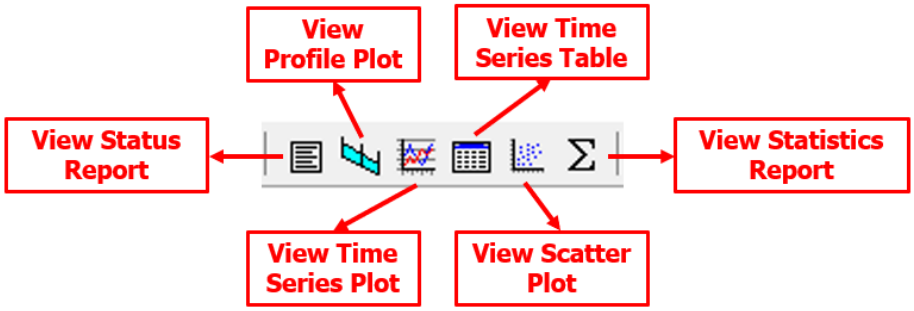


Figure 3-5. Viewing functions in SWMM

3.2. Reports in SWMM

The status report of an SWMM project can be viewed by clicking on the ***"Status Report"*** button (Figure 3-5). The results of the status report and summary of the report are shown in Figure 3-6a and Figure 3-6b, respectively.

(a) **(b)**

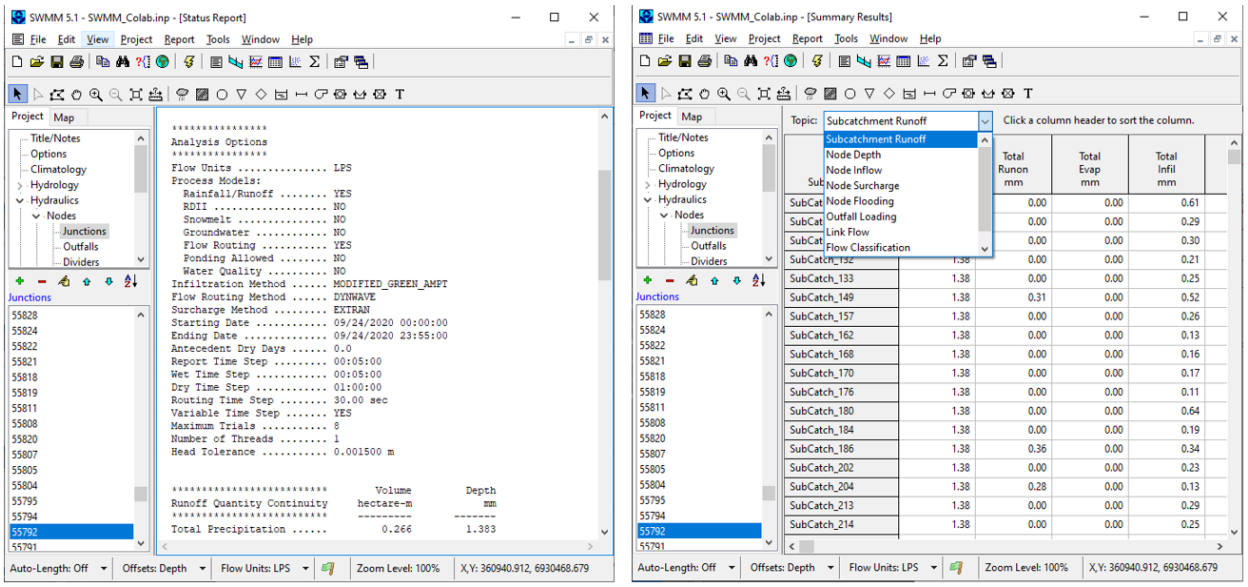


Figure 3-6. Reports in SWMM

SWMM clusters many properties of each component in the summary report. The user can click on the ***"Topic"*** combo box to see more options (Figure 3-6b).

3.3. Profile Plot in SWMM

A Profile plot displays the variation in simulated water depth with distance over a connected path of drainage system links and nodes at a particular point in time (Figure 3-7).

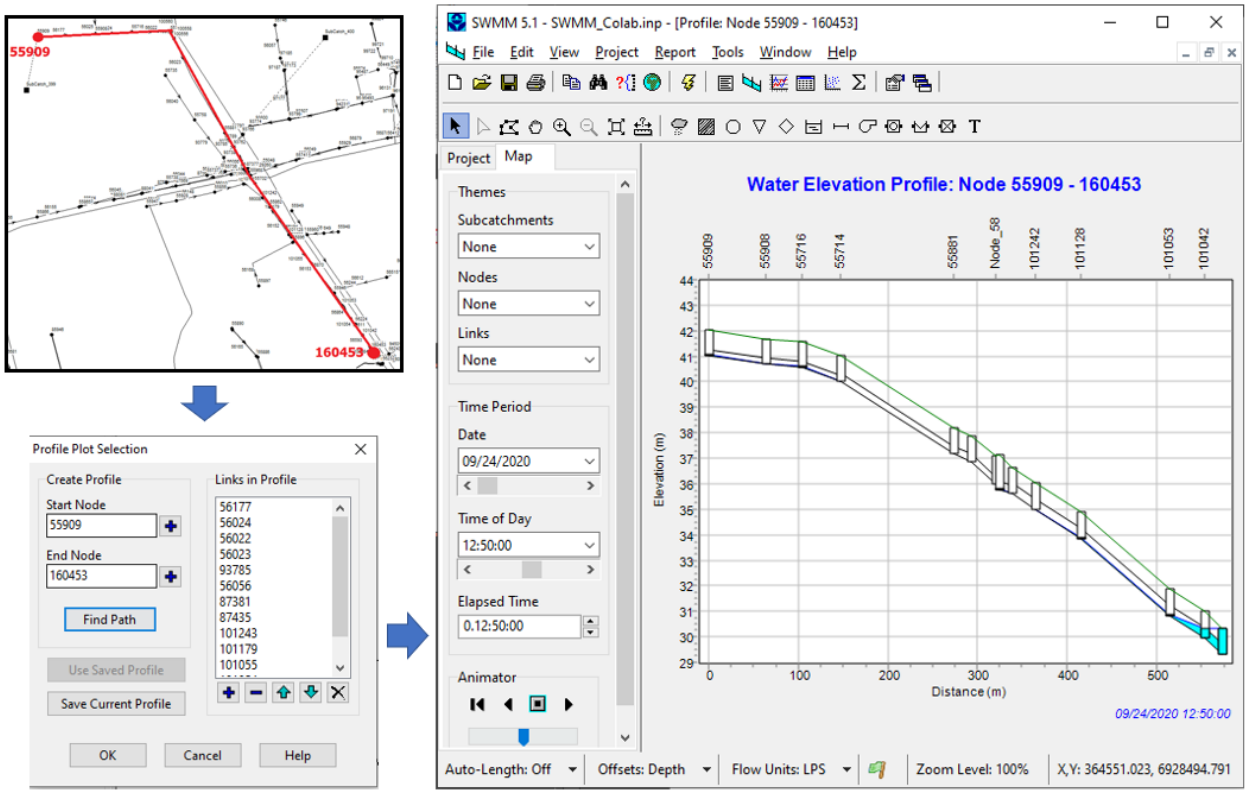


Figure 3-7. Profile plot in SWMM

3.4. Time Series Plots in SWMM

A Time Series Plot graphs the values over time of specific combinations of objects and variables (Figure 3-8).

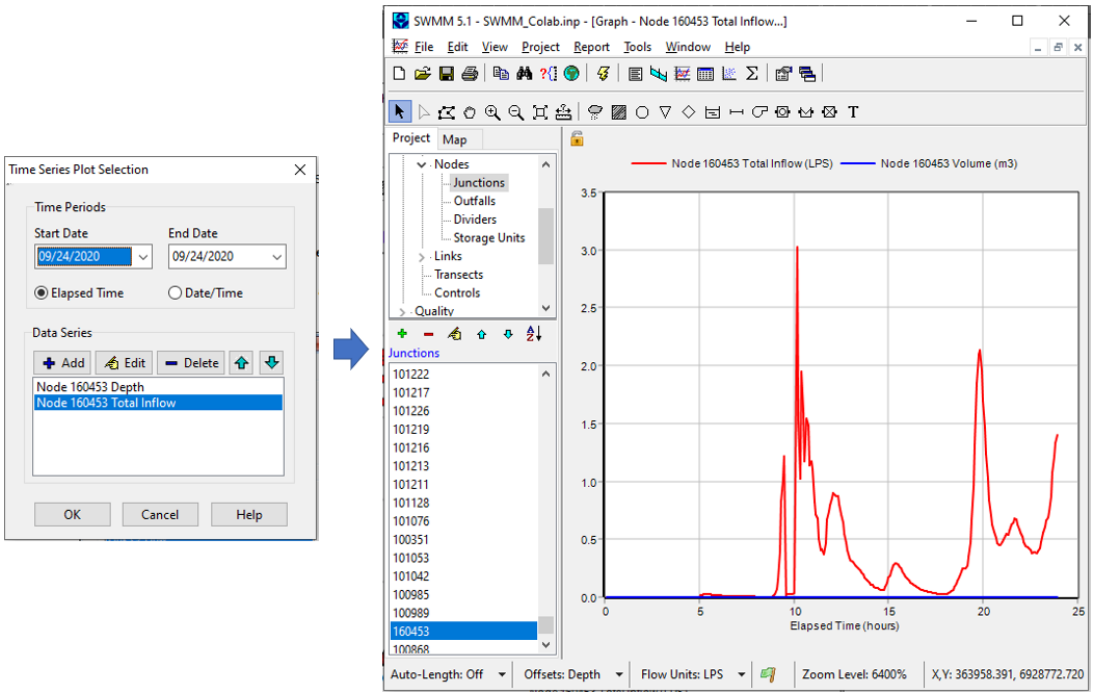
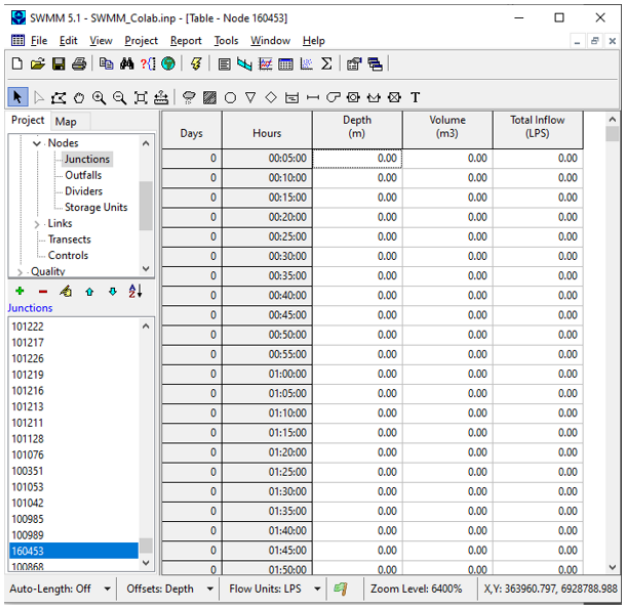


Figure 3-8. Time series plot in SWMM

A Time series results can be viewed in a tabular format. There are two types of table-based formats:

- *Table by Object*: tabulates the time series of several variables for a single object (e.g., flow and water depth for a conduit). This table format is used when creating a time series table of several variables for a single object (Figure 3-9a).
- *Table by Variable*: tabulates the time series of a single variable for several objects of the same type (e.g., runoff for a group of sub-catchments). This format is used when creating a time series table of a single variable for one or more objects (Figure 3-9b).

(a)



(b)

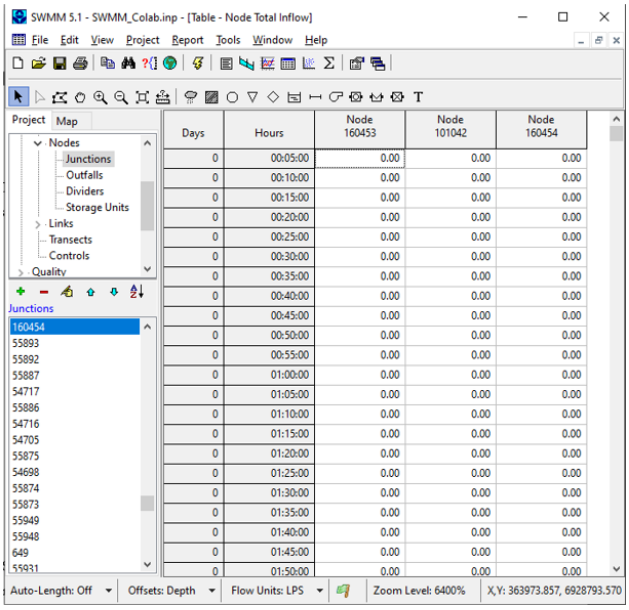


Figure 3-9. Time series tables in SWMM

3.5. Scatter Plot in SWMM

A Scatter Plot displays the relationship between a pair of variables, such as flow rate in a pipe versus water depth at a node (Figure 3-10).

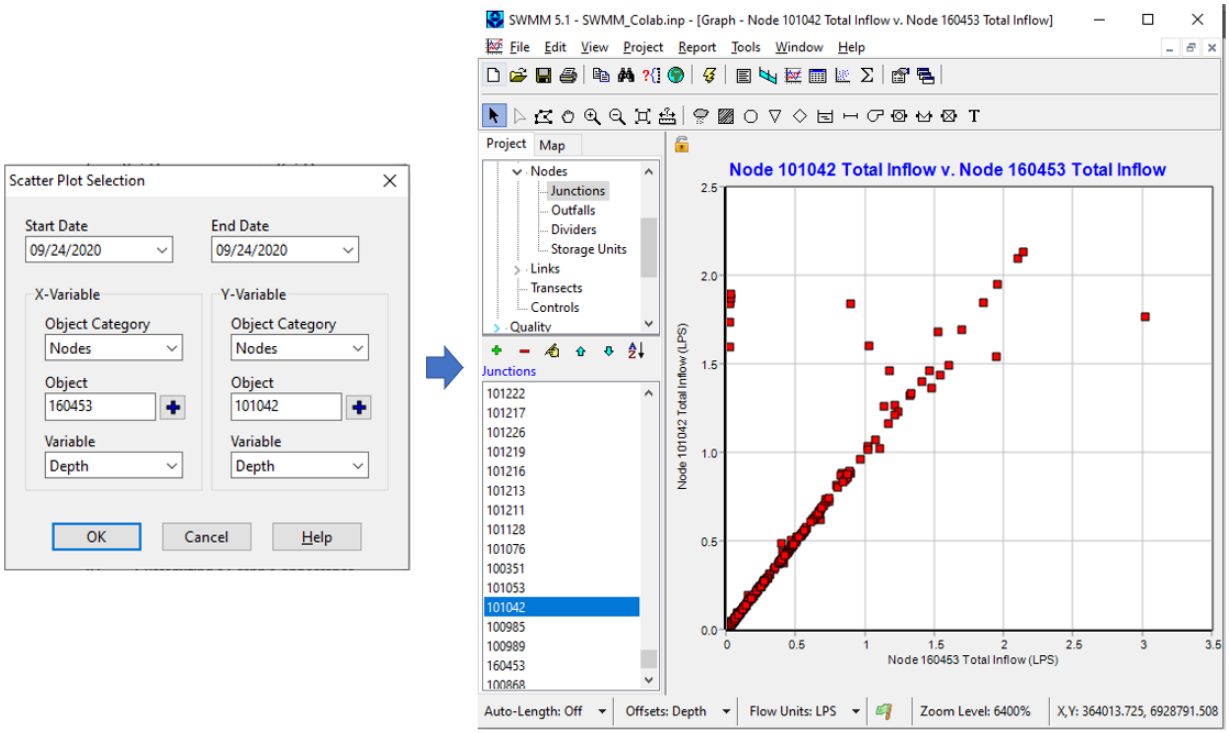


Figure 3-10. Scatter plot in SWMM

3.6. Statistics Reports in SWMM

A Statistics report will: 1) segregate the simulation period into a sequence of non-overlapping events, either by day, month, or by flow (or volume) above some minimum threshold value; 2) compute a statistical value that characterizes each event, such as the mean, maximum, or total sum of the variable over the event's time-period; 3) compute summary statistics for the entire set of event values (mean, standard deviation and skewness); and 4) perform a frequency analysis on the set of event values (Figure 3-11).

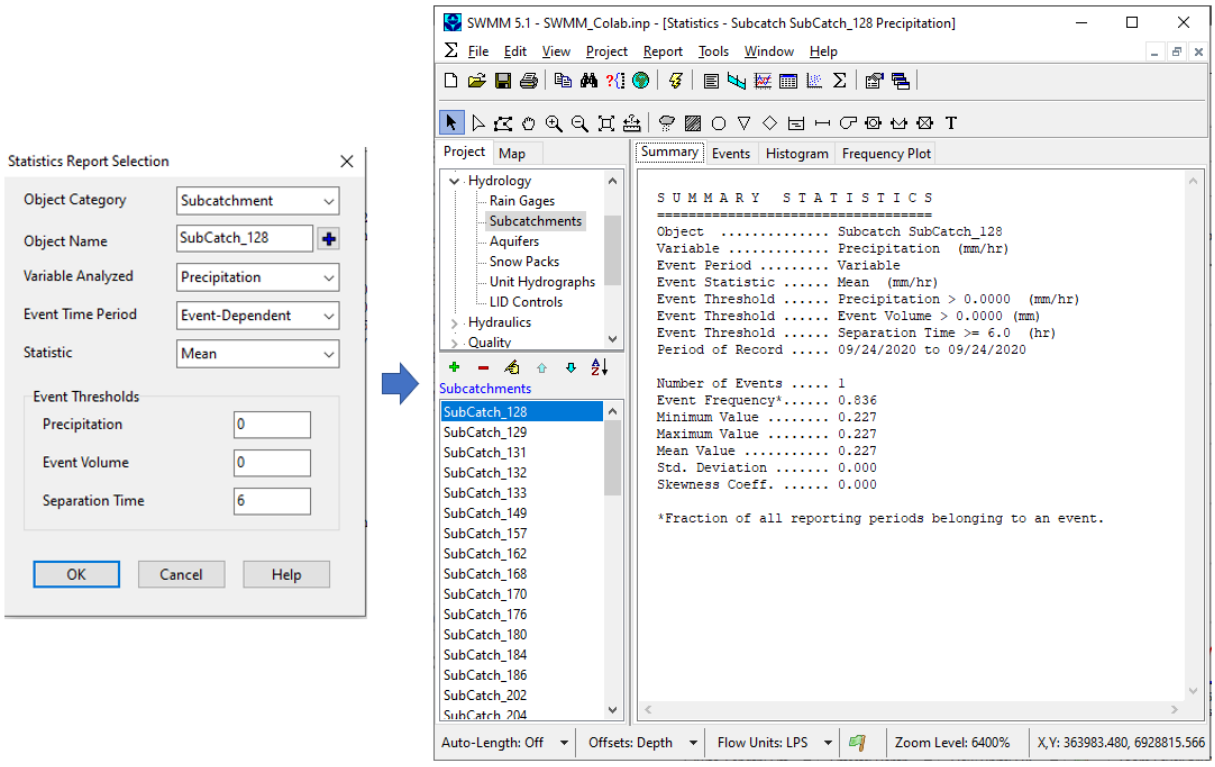


Figure 3-11. Statistics report in SWMM

4. PySWMM Simulation Package

4.1. Introduction to PySWMM

PySWMM is a Python language software package for the creation, manipulation, and study of the structure, dynamics, and function of complex networks. This package allows users to load and manipulate SWMM models. The PySWMM package can be used by engineers, modelers, and researchers who want to streamline stormwater modeling optimization, controls, and post-processing results.

PySWMM is intended to provide:

- Tools for the study of the structure and dynamics within USEPA SWMM5
- A standard programming interface and graph implementation that is suitable for many applications
- A rapid development environment for collaborative, multidisciplinary projects
- An interface to USEPA SWMM5
- Development and implementation of control logic outside of native EPA-SWMM Controls

- Methods for users to establish their node inflows
- A coding interface to binary output files
- New modeling possibilities for the SWMM5 Community

4.2. Installing Python Environment

Python environment is critical in running the PySWMM package. To install python, users can do it in several ways. The simplest way to install python is to use Anaconda:

- *Step 1:* Download Anaconda from the homepage address: https://www.anaconda.com/products/distribution

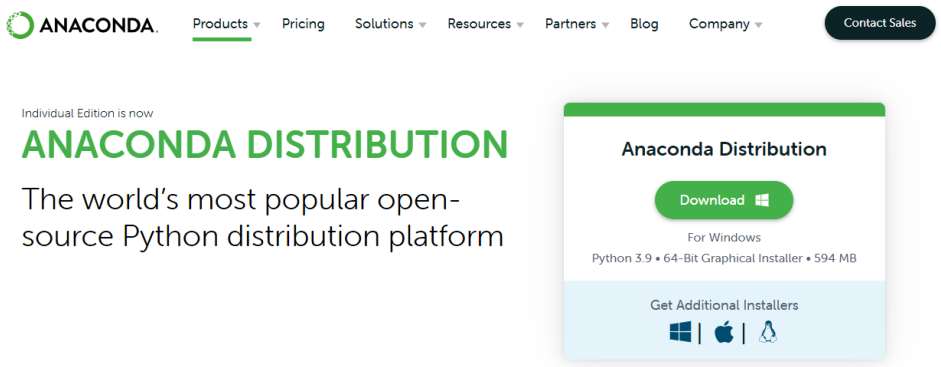


Figure 4-1. Anaconda homepage

- *Step 2:* Install Anaconda.

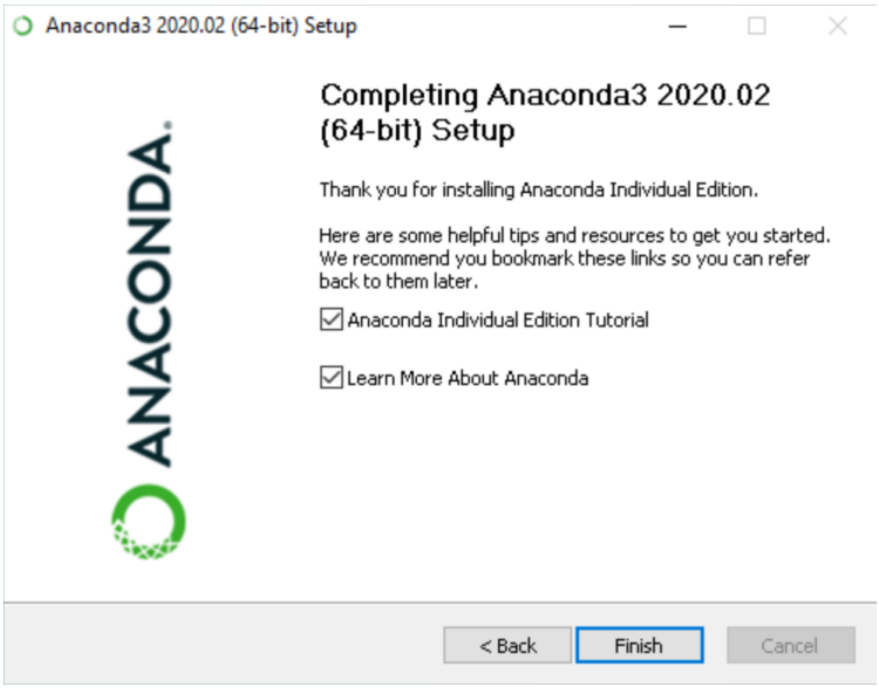


Figure 4-2. Installing Anaconda

4.3. Installing PySWMM via Anaconda

- *Step 1:* Start Anaconda Prompt from PC: *Start/Anaconda 3(64-bit)/Anaconda Prompt (Anaconda3)*

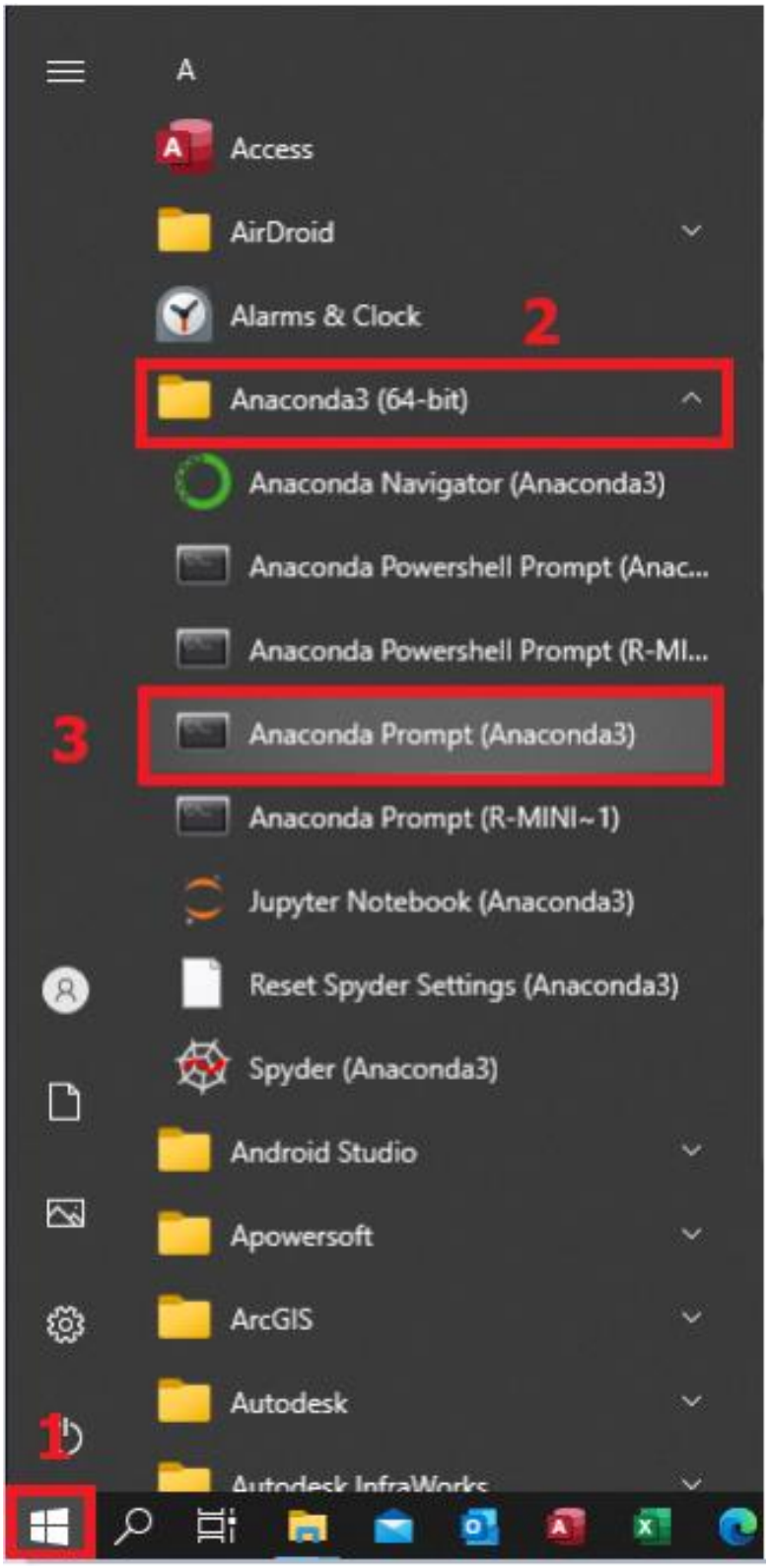


Figure 4-3. Window Prompt

➢ *Step 2:* Install PySWMM by typing the command: *pip install pyswmm*

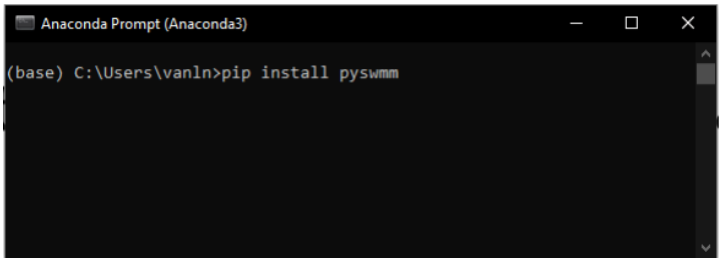


Figure 4-4. Installing PySWMM

➢ *Step 3:* Check PySWMM:

– Open Anaconda Prompt: Start/Anaconda 3(64-bit)/Anaconda Prompt (Anaconda3)

– Start python.

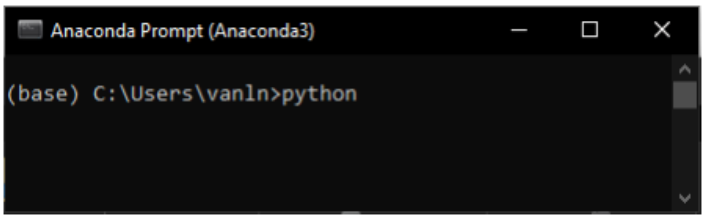


Figure 4-5. Starting python

– Import PySWMM and check:

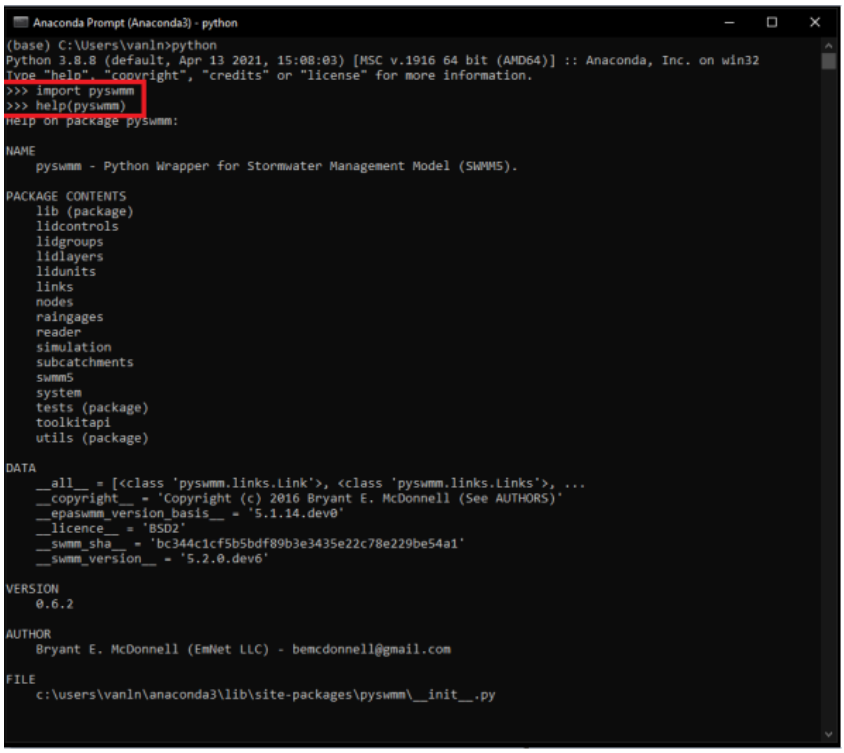


Figure 4-6. Checking PySWMM availability

4.4. PySWMM Examples

- Import necessary components in PySWMM

```
from pyswmm import Simulation
from pyswmm import Nodes
from pyswmm import Links
from pyswmm import Subcatchments
from pyswmm import RainGages
import matplotlib.pyplot as plt
```

- Specify the location of the SWMM project

```
data_file = "C:/Users/vanln/OneDrive - NTNU/Documents/DuLieu_CamXoa/PhDThesis/Prepar
#data_file = "SWMM_Project.inp"
```

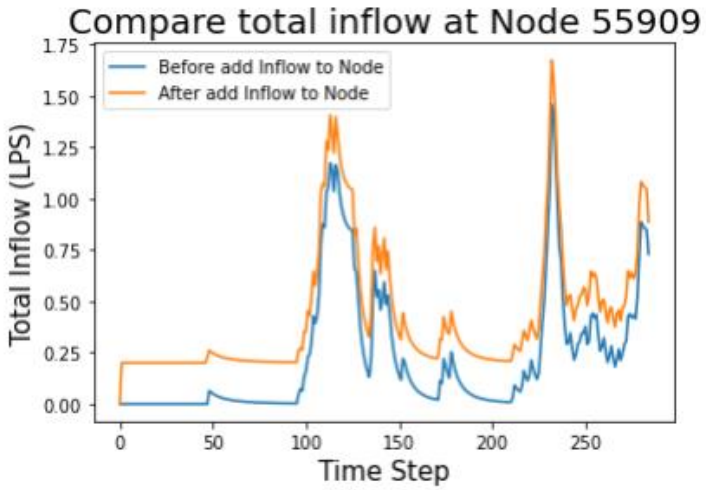
- Load and Run the SWMM model

```
# Load SWMM model, run, and get simulation current time
with Simulation(data_file) as sim:
    for step in sim:
        #print(sim.current_time.strftime('%m/%d/%Y %H:%M:%S'))
        pass
```

- Run the SWMM model and Get inflow at a specific node

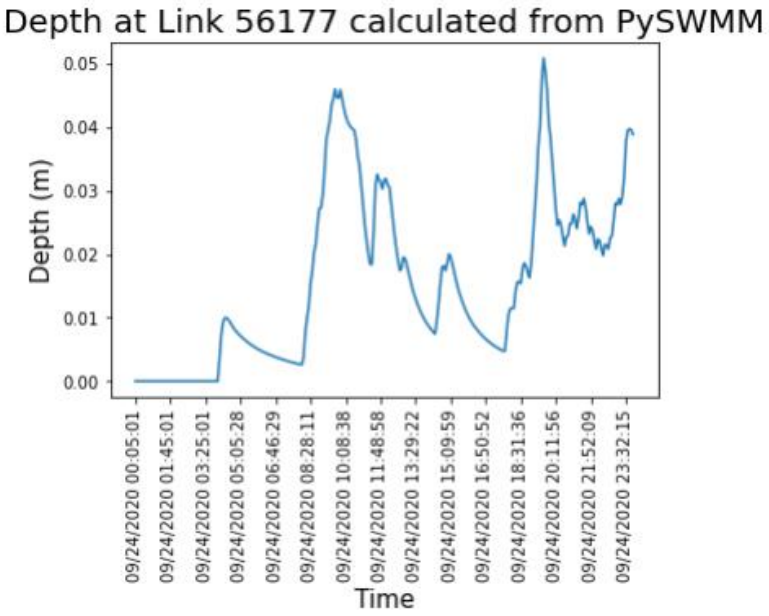
```
# Compare flow at Node '55909' before and after adding inflow

plt.plot(total_inflow, label='Before add Inflow to Node')
plt.plot(total_inflow_new, label='After add Inflow to Node')
#plt.xticks(n_new[::20], rotation=90)
plt.xlabel('Time Step', fontsize=15)
plt.ylabel('Total Inflow (LPS)', fontsize=15)
plt.title("Compare total inflow at Node " +  node_n, fontsize=20)
plt.legend()
plt.show()
```



➢ Run the SWMM model and Get depth at a specific link

```
plt.plot(n, depth)
plt.xticks(n[::20], rotation=90)
plt.xlabel('Time', fontsize=15)
plt.ylabel('Depth (m)', fontsize=15)
plt.title("Depth at Link " +  link_n + " calculated from PySWMM", fontsize=20)
plt.show()
```



Details of PySWMM commands have been listed in Appendix A.

Moreover, users can refer to PySWMM documentation for more details at this webpage: https://pyswmm.readthedocs.io/en/latest/ or GitHub page: https://github.com/OpenWaterAnalytics/pyswmm.

4.5. Advantages and Disadvantages

❖ ***Advantages***

– Can execute SWMM model automatically

– Can design many SWMM scenarios (by changing), run, and get results

– Can be used to calibrate the SWMM model automatically (trails and errors approach)

– Can adjust many properties of components for multi-simulation:

- Node: Inflow, Invert Elevation, Max. Depth, Initial Depth, Surcharge Depth, Ponded Area
- Link: Inlet/Outlet Offset, Initial Flow, Maximum Flow, Entry/Exit/Avg. Loss Coefficient
- SubCatchment: Area, Width, %Slope

❖ ***Disadvantages***

- Time steps are not the same, difficult for comparing simulated values to measured values (interpolation can be used)
- Cannot change some properties of components for multi-simulation:
  - Link: Shape, Max. Depth, Roughness
  - SubCatchment: Outlet, Rain Gage, %Imperviousness, NImperviousness, N-Perviousness, DStore-Imperviousness, DStorePerviousness

References

1. Rossman, L. Storm Water Management Model User's Manual Version 5.1 – manual (2015). U.S. EPA Office of Research and Development, Washington, DC, EPA/600/R-14/413 (NTIS EPA/600/R-14/413b).
2. QGIS Development Team (2022). QGIS Geographic Information System. Open Source Geospatial Foundation Project. http://qgis.osgeo.org.
3. Anon, (2020). Anaconda Software Distribution, Anaconda Inc. Available at: https://docs.anaconda.com/.
4. McDonnell, Bryant E., Ratliff, Katherine M., Tryby, Michael E., Wu, Jennifer Jia Xin, & Mullapudi, Abhiram. (2020). PySWMM: The Python Interface to Stormwater Management Model (SWMM). Journal of Open Source Software, 5(52), 2292, https://doi.org/10.21105/joss.02292.

Appendix A: PySWMM Commands

❖ Load SWMM model:

```
>>> from pyswmm import Simulation
>>>
>>> with Simulation('./testmodel.inp') as sim:
...     for step in sim:
...             pass
```

❖ Nodes

1) *Load Nodes*

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('./testmodel.inp') as sim:
...     node_object = Nodes(sim)
...
...     #J1 node instantiation
...     J1 = node_object["J1"]
...     print(J1.invert_elevation)
...     print(J1.is_junction())
...
...     #Step through a simulation
...     for step in sim:
...             print(J1.total_inflow)
...
```

2) *Generate Node Inflows*

```
>>> with Simulation('/testmodel.inp') as sim:
...      j1 = Nodes(sim)["J1"]
...      for step in sim:
...              j1.generated_inflow(9)
```

3) *Get/Set Node Full Depth*

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     j1 = Nodes(sim)["J1"]
...     print j1.full_depth
...     j1.full_depth = 50
...     print j1.full_depth
```

4) *Get/Set Node Invert Elevation*

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     j1 = Nodes(sim)["J1"]
...     print j1.invert_elevation
...     j1.invert_elevation = 0.2
...     print j1.invert_elevation
>>> 0.1
>>> 0.2
```

5) *Get/Set Node Initial Depth*

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     j1 = Nodes(sim)["J1"]
...     print j1.initial_depth
...     j1.initial_depth = 1
...     print j1.initial_depth
>>> 0
>>> 1
```

6) *Get/Set Node Surcharge Depth*

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     j1 = Nodes(sim)["J1"]
...     print j1.surcharge_depth
...     j1.surcharge_depth = 50
...     print j1.surcharge_depth
>>> 10
>>> 50
```

7) *Get/Set Node Ponding Area*

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     j1 = Nodes(sim)["J1"]
...     print j1.ponding_area
...     j1.ponding_area = 50
...     print j1.ponding_area
>>> 0
>>> 50
```

8) *Get Node Results for Depth*

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     j1 = Nodes(sim)["J1"]
...     for step in sim:
...         print j1.depth
>>> 0
>>> 0.5
```

9) Get Node Results for Head

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     j1 = Nodes(sim)["J1"]
...     for step in sim:
...         print j1.head
>>> 10
```

10) Get Node Results for Flooding Rate

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     j1 = Nodes(sim)["J1"]
...     for step in sim:
...         print j1.flooding
>>> 0
```

11) Get Node Results for Total Inflow Rate

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     j1 = Nodes(sim)["J1"]
...     for step in sim:
...         print j1.total_inflow
>>> 0
```

12) Get Node Results for Total Outflow Rate

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     j1 = Nodes(sim)["J1"]
...     for step in sim:
...         print j1.total_outflow
>>> 0
```

13) Get Node Results for Volume

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     j1 = Nodes(sim)["J1"]
...     for step in sim:
...         print j1.volume
>>> 0
```

14) Get Node Results for Lateral Inflow Rate

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     j1 = Nodes(sim)["J1"]
...     for step in sim:
...         print j1.lateral_inflow
>>> 0
>>> 0.25
```

15) Get Node Results for Losses Rate (Evaporation and Exfiltration)

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     j1 = Nodes(sim)["J1"]
...     for step in sim:
...         print j1.losses
>>> 0
>>> 0.01
```

16) Check if Node is a Divider Type

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     j1 = Nodes(sim)["J1"]
...     print j1.is_divider()
>>> True
```

17) Check if Node is a Junction Type

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     j1 = Nodes(sim)["J1"]
...     print j1.is_junction()
>>> True
```

18) Check if Node is an Outfall Type

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     j1 = Nodes(sim)["J1"]
...     print j1.is_outfall()
>>> True
```

19) Check if Node is a Storage Type

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     j1 = Nodes(sim)["J1"]
...     print j1.is_storage()
>>> True
```

20) Get Current Water Quality Values for a Node

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('tests/buildup-test.inp') as sim:
...     J1 = Nodes(sim)["J1"]
...     for step in sim:
...         print(J1.pollut_quality)
```

❖ Links

1) Load Links

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     for link in Links(sim):
...         print link
...         print link.linkid
```

2) Get/set Average Link Loss

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     print c1c2.average_head_loss
...     c1c2.average_head_loss = 0.2
...     print c1c2.average_head_loss
>>> 0
>>> 0.2
```

3) Get/Set Link Flow Limit

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     print c1c2.flow_limit
...     c1c2.flow_limit = 0.2
...     print c1c2.flow_limit
>>> 0
>>> 0.2
```

4) Get/Set Link Initial Flow

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     print c1c2.initial_flow
...     c1c2.initial_flow = 0.2
...     print c1c2.initial_flow
>>> 0.1
>>> 0.2
```

5) Get/Set Inlet Head Loss

```
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     print c1c2.inlet_head_loss
...     c1c2.inlet_head_loss = 0.2
...     print c1c2.inlet_head_loss
>>> 0
>>> 0.2
```

6) Get/Set Outlet Head Loss

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     print c1c2.outlet_head_loss
...     c1c2.outlet_head_loss = 0.2
...     print c1c2.outlet_head_loss
>>> 0
>>> 0.2
```

7) Get/set Upstream Offset Depth

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     print c1c2.inlet_offset
...     c1c2.inlet_offset = 0.2
...     print c1c2.inlet_offset
>>> 0.1
>>> 0.2
```

8) Get/Set Downstream Offset Depth

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     print c1c2.outlet_offset
...     c1c2.outlet_offset = 0.2
...     print c1c2.outlet_offset
>>> 0.1
>>> 0.2
```

9) Get/Set Link Seepage Loss

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     print c1c2.seepage_rate
...     c1c2.seepagerate = 0.2
...     print c1c2.seepage_rate
>>> 0
>>> 0.2
```

10) Get/Set Link Target Setting

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     for step in sim:
...         print c1c2.target_setting
...         if c1c2.flow > 3:
...             c1c2.target_setting = 0.1
>>> 0
>>> 0
>>> 0.1
```

11) Get Link Results for Depth

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     for step in sim:
...         print c1c2.depth
>>> 0
>>> 1.2
```

12) Get Link Results for Flow

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     for step in sim:
...         print c1c2.flow
>>> 0
>>> 1.2
```

13) Get Link Results for Volume

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:

...     c1c2 = Links(sim)["C1:C2"]
...     for step in sim:
...         print c1c2.volume
>>> 0
>>> 1.2
```

14) Get Link Upstream and Downstream Node IDs

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     print c1c2.connections
>>> ("C1","C2")
```

15) Get Link Current Setting

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     for step in sim:
...         print c1c2.current_setting
>>> 0
>>> 1
```

16) Get Link Results for Froude

```
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     for step in sim:
...         print c1c2.froude
>>> 0
```

17) Check if Link is a Conduit Type

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     print c1c2.is_conduit()
>>> True
```

18) Check if Link is an Orifice Type

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     print c1c2.is_orifice()
>>> False
```

19) Check if Link is an Outlet Type

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     print c1c2.is_outlet()
>>> False
```

20) Check if Link is a Pump Type

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     print c1c2.is_pump()
>>> False
```

21) Check if Link is a Weir Type

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     print c1c2.is_weir()
>>> False
```

22) Get Current Water Quality Values for a Link

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/buildup-test.inp') as sim:
...     C1 = Nodes(sim)["C1"]
...     for step in sim:
...         print(C1.pollut_quality)
>>> {'test-pollutant': 0.0}
```

23) Get Total Pollutant Loading Values for a Link

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/buildup-test.inp') as sim:
...     C1 = Nodes(sim)["C1"]
...     for step in sim:
...         print(C1.total_loading)
>>> {'test-pollutant': 0.01}
```

24) Get Link ID

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     print c1c2.linkid
>>> "C1"
```

25) Get Link Inlet Node ID

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     print c1c2.inlet_node
>>> C1
```

26) Get Link Outlet Node ID

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     print c1c2.outlet_node
>>> C2
```

❖ Subcatchments

1) *Load Subcatchments*

```
>>> from pyswmm import Simulation, Subcatchments
>>>
>>> with Simulation('./testmodel.inp') as sim:
...     subcatch_object = Subcatchments(sim)
...
...     #SC1 subcatchment instantiation
...     SC1 = subcatch_object["S1"]
...     print(SC1.area)
...
...     #Step through a simulation
...     for step in sim:
...             print(SC1.runoff)
...
```

2) *Get/Set Subcatchment Area*

```
>>> from pyswmm import Simulation, Subcatchments
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     s1 = Subcatchments(sim)["S1"]
...     print s1.area
...     s1.area = 50
...     print s1.area
>>> 10
>>> 50
```

3) *Get/Set Subcatchment Width*

```
>>> from pyswmm import Simulation, Subcatchments
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     s1 = Subcatchments(sim)["S1"]
...     print s1.width
...     s1.width = 30
...     print s1.width
>>> 100
>>> 30
```

4) *Get/Set Subcatchment Slope*

```
>>> from pyswmm import Simulation, Subcatchments
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     s1 = Subcatchments(sim)["S1"]
...     print s1.slope
...     s1.slope = 0.02
...     print s1.slope
>>> 0.1
>>> 0.2
```

5) Get/Set Subcatchment Percent Impervious

```
>>> from pyswmm import Simulation, Subcatchments
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     s1 = Subcatchments(sim)["S1"]
...     print s1.percent_impervious
...     s1.percent_impervious = 50
...     print s1.percent_impervious
>>> 10
>>> 50
```

6) Get/Set Subcatchment Curb Length

```
>>> from pyswmm import Simulation, Subcatchments
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     s1 = Subcatchments(sim)["S1"]
...     print s1.curb_length
...     s1.curb_length = 100
...     print s1.curb_length
>>> 0
>>> 100
```

7) Get Pollutant Results for Surface Buildup

```
>>> from pyswmm import Simulation, Subcatchments
>>>
>>> with Simulation('tests/buildup-test.inp') as sim:
...     s1 = Subcatchments(sim)["S1"]
...     for step in sim:
...         print(s1.buildup)
>>> {'test-pollutant': 8.0}
```

8) Get Pollutant Results for Current Concentration of Pollutant in Ponded Water

```
>>> from pyswmm import Simulation, Subcatchments
>>>
>>> with Simulation('tests/buildup-test.inp') as sim:
...     s1 = Subcatchments(sim)["S1"]
...     for step in sim:
...         print(s1.conc_ponded)
>>> {'test-pollut1': 0.0, 'test-pollut2': 0.0}
```

9) Get Subcatchment Results for Evaporation Loss

```
>>> from pyswmm import Simulation, Subcatchments
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     s1 = Subcatchments(sim)["S1"]
...     for step in sim:
...         print s1.evaporation_loss
>>> 0.01
```

10) Get Subcatchment Results for Infiltration Loss

```
>>> from pyswmm import Simulation, Subcatchments
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     s1 = Subcatchments(sim)["S1"]
...     for step in sim:
...         print s1.infiltration_loss
>>> 0
>>> 0.01
```

11) Get Current Pollutant Water Quality Results from Subcatchment Runoff

```
>>> from pyswmm import Simulation, Subcatchments
>>>
>>> with Simulation('tests/buildup-test.inp') as sim:
...     s1 = Subcatchments(sim)["S1"]
...     for step in sim:
...         print(s1.pollut_quality)
```

12) Get Subcatchment Results for Rainfall

```
>>> from pyswmm import Simulation, Subcatchments
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     s1 = Subcatchments(sim)["S1"]
...     for step in sim:
...         print s1.rainfall
>>> 0
>>> 1.2
```

13) Get Subcatchment Results for Runoff Rate

```
>>> from pyswmm import Simulation, Subcatchments
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     s1 = Subcatchments(sim)["S1"]
...     for step in sim:
...         print s1.runoff
>>> 0
```

14) Get Total Pollutant Loading from Subcatchment Runoff

```
>>> from pyswmm import Simulation, Subcatchments
>>>
>>> with Simulation('tests/buildup-test.inp') as sim:
...     s1 = Subcatchments(sim)["S1"]
...     for step in sim:
...         print(s1.runoff_total_loading)
>>> {'TSS': 0.01, 'Lead': 0.001}
```

15) Get Subcatchment Results for Run On

```
>>> from pyswmm import Simulation, Subcatchments
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     s1 = Subcatchments(sim)["S1"]
...     for step in sim:
...         print s1.runon
>>> 0
>>> 1.2
```

16) Get Subcatchment Results for Snow Depth

```
>>> from pyswmm import Simulation, Subcatchments
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     s1 = Subcatchments(sim)["S1"]
...     for step in sim:
...         print s1.snow_depth
>>> 0
```