Copyright

by

Edward Davis Tiernan

2018

The Thesis Committee for Edward Davis Tiernan

Certifies that this is the approved version of the following thesis:

Developing SWMMCALPY: an automated, genetic approach to calibrating the storm water management model

APPROVED BY

SUPERVISING COMMITTEE:

Ben R. Hodges, Supervisor

Lina Sela

Developing SWMMCALPY: an automated, genetic approach to calibrating the storm water management model

by

Edward Davis Tiernan

Thesis

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN ENGINEERING

The University of Texas at Austin

May 2018

Acknowledgments

This thesis was developed under Cooperative Agreement No. 83595001 awarded by the U.S. Environmental Protection Agency to The University of Texas at Austin. It has not been formally reviewed by EPA. The views expressed in this document are solely those of Edward Davis Tiernan and do not necessarily reflect those of the Agency. EPA does not endorse any products or commercial services mentioned in this publication.

The author thanks both Geosyntec Consultants, Inc. and the City of Austin for the datasets used to produce the test case in Chapter 3

Developing SWMMCALPY: an automated, genetic approach to calibrating the storm water management model

Edward Davis Tiernan, M.S.E.

The University of Texas at Austin, 2018

Supervisor: Ben R. Hodges

Parameter calibration is considered a crucial, albeit arduous, step for reliable performance of the Storm Water Management Model (SWMM) that engineers often undertake manually. This research presents an open-source, automated calibration routine that returns a calibrated model input file to the user. The routine first represents the catchment network as a directed graph object using the NetworkX python package for flexibility in handling real-world observed data availability. Once the calibratable subset of the system is identified, a multi-objective, genetic algorithm (modified Non-dominated Sorting Genetic Algorithm II: NSGA-II) estimates the Pareto front for the objective functions within the feasible performance space. The solutions on this Pareto front represent the optimized parameter sets for matching simulated and observed catchment behavior. A specific solution among this Pareto set can be chosen by assigning weights to the objective functions. This solution is then returned to the user, completing a fully automated process that requires minimal user input and does not require intervention or selections during calibration.

Keywords: SWMM, Automated Calibration, NetworkX, Genetic Algorithm, NSGA-II, Multi-objective Function Optimization

Table of Contents

List of Tables

List of Figures

Acronyms

NED Normalized Euclidean Distance. x, 29, 30, 33, 34, 58, 59

NPE Normalized Peak Error. x, 29, 30, 32, 34, 52, 55, 59

NSE_m modified Nash-Sutcliffe Efficiency. x, 29, 30, 32, 34, 55, 59

NSE Nash-Sutcliffe Efficiency index. 17, 31, 32, 52, 53, 71–73, 75–77, 80

NVE Normalized Volume Error. x, 30, 32, 34, 55, 58, 59

ANN Artificial Neural Networks. 10, 75, 77

EPA Environmental Protection Agency. x, xi, 1–3, 6, 9, 12, 20, 21, 25, 46, 48, 67, 89

EXTRAN External Transport. 4

GIS Geographic Information System. 6, 22, 50, 70

GLUE Generalized Likelihood Uncertainty Estimation. 63, 77–81, 83

HEC-HMS Hydrologic Engineering Center - Hydrologic Modeling System. 5, 67

IDF Intensity-Duration-Frequency curves. 1

LID Low Impact Design. 71

MOGSA Multi-Objective Generalized Sensitivity Analysis. 79, 81

NCIMM National Center for Infrastructure Modeling and Management. 2, 6

NSGA-II Non-dominated Sorting Genetic Algorithm - II. v, ix, x, 2, 10, 12, 17–19, 28, 29, 33–37, 40, 42–45, 52, 56, 57, 59, 61, 62, 64, 74, 76–78, 82, 128, 130, 132–134

OWA Open Water Analytics. 6, 11

RMSE Root Mean Squared Error. 31

SCS Soil Conservation Service. 1

SRTC Sensitivity-based Ratio Tuning Calibration. 2, 9, 17, 32, 33, 49, 50, 55, 144

SWAT Soil and Water Assessment Tool. 5

SWMM Storm Water Managment Model. v, vi, x, xi, 1–13, 15, 17, 19, 20, 23–26, 28, 33, 36, 38, 40, 41, 43, 46, 48–52, 54, 59–64, 67–69, 74, 76, 77, 81–84, 87, 89, 102, 144

SWMM-UI Storm Water Management Model User Interface. 2, 6, 12

Chapter 1: Introduction

1.1 Problem Statement

Urban storm water runoff assessments have been conducted as part of urban planning for nearly as long as cities have been planned (Angelakis, 2017). Empirical methods such as the rational method, unit hydrographs, and intensity-duration frequency (IDF) curves were used for many years (Baiamonte & Singh, 2017; Wang, Liu, & Yang, 2012) with individual institutions implementing their own estimation methods, such as the United States Soil Conservation Service's (SCS) curve number method (Mishra & Singh, 2003). These methods are empirical, black box approaches that consider the catchment under scrutiny to be an opaque control volume with precipitation as the input and runoff as the output. A unit hydrograph, for example, relates a hyetograph directly to the marginal runoff and a simple convolution will yield the storm-specific hydrograph (Wang et al., 2012). As technology and computing power advanced, engineers broke open the black box, taking on the physically-based equations governing hydrological processes, such as the Saint-Venant and Richards Equations that had been widely known but challenging at large scales (e.g Audusse, Bouchut, Bristeau, Klein, & Perthame, 2004; Lappala, Healy, & Weeks, 1987; Liang & Marche, 2009). A caveat for the physically-based model approach, however, is an increased number of necessary parameters to inform the governing equations.

This thesis is a detailed report on the development of SWMMCALPY. SWMMCALPY supports the EPAs Storm Water Management Model (SWMM) as

an automated calibration tool written in the python coding language. The name, SWMMCALPY, comes from the purpose and composition of the tool, SWMM CAL-ibration using PY-thon. SWMM, as a physically-based hydrologic-hydraulic model, contains several parameters that require calibration for reliable model performance (Rossman et al., 2010). SWMMCALPY seeks to calibrate these parameter values to optimize model simulation fit to observed data without the need for direct user interference. The Non-dominated Sorting Genetic Algorithm (NSGA-II) is employed to solve the optimization problem in an automated fashion. SWMMCALPY will be coupled with the SWMM User Interface (SWMM-UI) and released as open source to advance the directive of the National Center for Infrastructure Modeling and Management (NCIMM) (NCIMM, 2018). According to Rossman (Rossman et al., 2010), there is currently no automated calibration procedure recommended by the EPA for SWMM version 5.

Contemporaneously, a test case involving the Brentwood system in north Austin was prepared as a more compelling evaluation of the performance of SWMM-CALPY. The hypothesis is that SWMMCALPYs method will produce calibrated solution that performs comparably to that of PCSWMMs Sensitivity-based Radio Tuning Calibration SRTC tool.

Chapter 1 presents a necessary overview of the problem statement SWMM-CALPY seeks to address. Choice components of SWMM itself are reviewed to introduce hydrologic-hydraulic concepts that are referred to heavily in later chapters. A brief comparison of SWMM to other common hydrologic models helps highlight the value provided by SWMMCALPY to the world of water resources engineering. Subsequently, a literature review of past work relating to SWMM calibration and the use of python pertaining to SWMM innovation is presented.

Finally, the objectives for SWMMCALPY are enumerated explicitly and broken down into a format that offers a natural outline for the methodology in Chapter 2.

1.2 Introduction to SWMM

In the early 1970s, the U.S. Environmental Protection Agency (EPA) began development of SWMM as one of the first hydrologic-hydraulic solvers (Huber & Roesner, 2012). The overarching purpose was to aid in planning, designing, and analyzing storm water runoff quantity as well as quality in urban drainage networks (Rossman & Huber, 2016). Since the first release of SWMM, the model has undergone five major updates, including the most recent, in 2005, which was a complete rewrite of the SWMM numerical solver in C code (Huber & Roesner, 2012; Rossman et al., 2010).

SWMM is a combined hydrologic-hydraulic model, meaning that the surface hydrology informs the channel/pipe routing of the drainage network (Rossman et al., 2010). The model requires forcing data in the form of timeseries rainfall information in order to produce runoff. It can also accept other forcing data such as temperature and sun angle, as well as initial/boundary conditions for dry weather flow. All precipitation falls on subcatchment objects in SWMM, where the subcatchment is defined as the area that contributes to a single location, the outlet. SWMM conceptualizes each subcatchment to be a rectangle with area-averaged parameters (Rossman & Huber, 2016). Some of these parameters are physically real, such as Manning's coefficient of impervious surfaces, while others are intended to be model tuning parameters, such as the imposed uniform slope and width. The behavior of precipitation on the subcatchment is modeled by

the nonlinear reservoir model, which, for each time step, accounts for infiltration, evaporation, and runoff. The roughness component to the runoff is derived from Manning's equation, thus the importance of subcatchment slope and width. All runoff flows to the subcatchment outlet, where it is routed into SWMMs hydraulic network (Rossman & Huber, 2016).

Composed of common storm water management features like conduits, weirs, pumps, and orifices, SWMMs routing method utilizes the one-dimensional Saint-Venant equation, either in its full form or in the diffusive wave or kinematic wave approximations. The hydraulic solver uses an implicit backward's Euler scheme to solve for the transport of water within the hydraulic network (Rossman & Huber, 2017). The Saint-Venant equation was first implemented in the SWMM 3 release in 1981 as an additional extended transport (EXTRAN) module and is based on a link-node approach (Rossman et al., 2010). This link-node approach essentially reduces the complexity of the model, as computing the exchange of water is limited to occurring at the interface of links and nodes (Rossman & Huber, 2017).

SWMM is capable of solving for both water quality and quantity, which enables it to be used in a wide variety of applications (e.g. Alamdari, Sample, Steinberg, Ross, & Easton, 2017; Cipolla, Maglionico, & Stojkov, 2016; Guan, Sillanpää, & Koivusalo, 2015a). A full description of SWMMs processes and workflow is described in four manuals (hydrology (Rossman & Huber, 2016), hydraulics (Rossman & Huber, 2017), water quality (Rossman, 2016), and the user's manual (Rossman et al., 2010)).

Despite being a comprehensive and effective storm water runoff model (Rossman et al., 2010), SWMM does not exist in a vacuum. Many other models compete with SWMM or have slightly different focuses. SWMM is primarily used as an

urban drainage model (Guan, Sillanpää, & Koivusalo, 2015b); for more rural applications, a common tool is the Soil and Water Assessment tool (SWAT). SWAT was built and is maintained by a research team at Texas A&M (Ignatius & Jones, 2018). SWAT can account for a wide variety of land use types that can affect the hydrology of a system (Neitsch, Arnold, Kiniry, & Williams, 2011), juxtaposed against SWMMs pervious/impervious binary land use accounting (Rossman & Huber, 2016). The Hydrologic Engineering Center – Hydrologic Modeling System (HEC-HMS) developed by the US Army Core of Engineers, is another commonly used hydrologic model (Engineers, 2016; Srinivas, Singh, & Deshmukh, 2018). HEC-HMS boasts a bevy of transform hydrographs that convert rainfall into runoff without assuming a subcatchment shape (Engineers, 2016). Often coupled with the Hydrologic Engineering Center – River Analysis System, the HEC suite of programs is designed for river systems and larger, natural watersheds.

Something that SWMM has in common with SWAT and HEC-HMS is that they are all considered "clustered" or "lumped" models (Engineers, 2016; Neitsch et al., 2011; Rossman et al., 2010). This means that the forcing information is received by a collection of subcatchments, each with area-averaged parameter values that have the effect of smoothing out any higher resolution local features of the system. Not all models make this simplifying assumption. High resolution models like GSSHA and ADHYDRO create a fine scale grid that overlays the watershed (Downer & Ogden, 2004). This grid can be adjusted to accommodate features like curbs or building edges (Downer & Ogden, 2004). In a sense, each grid cell behaves similarly to a subcatchment in a clustered model (in far greater numbers) and greater inter-grid exchange of surface water. These "distributed" models, often maintain separate hydrologic and hydraulic schemes, but have the

potential for purely 2-dimensional hydrologic model by accounting for channel shape by use of extremely high-resolution grid cell deployment (Bao, Wang, Zhang, & Li, 2017).

Even within the SWMM community there are branches and competing software. The original EPA SWMM remains a popular open source option for engineers and researchers (EPA, 2014). Meanwhile, proprietary packages such as PCSWMM and XPSWMM offer additional functionality such as coupled GIS tools and time-saving calibration methods (CHI Water, 2018). In the open source community, the EPA has been providing maintenance and access to SWMM for decades (EPA, 2014). Other groups, like Open Water Analytics (OWA) out of Cincinnati, OH and the National Center for Infrastructure Modeling and Management (NCIMM) based in Austin, TX contribute to free projects. Examples of open source projects are OWA-SWMM and the SWMM-UI, both of which leverage the python coding language to make SWMMs workflow more transparent and accessible to the user (McDonnell, 2018a; NCIMM, 2018). The NCIMM directive is, in part, to bring about the next generation of SWMM, with a special emphasis on automating time-consuming aspects of the SWMM workflow, and building a faster solver for SWMMs hydraulics module, thus enabling the viability of larger models (NCIMM, 2018). SWMMCALPYs objective of returning a calibration parameter set solution to optimize model agreement with observations, without the need for user interference, contributes directly to NCIMMs goals for the future of SWMM.

1.3 Introduction to Calibration & SWMM

1.3.1 The Need for Calibrating SWMM

Appendix A contains a literature review of studies that undertake SWMM calibration and an analysis about which methods show the most promise for SWMM-CALPYs automated approach. Important points from Appendix A are summarized here.

SWMM is a physically-based model. SWMMs transparent control volume approach attempts to make predictions about the commonly sought outcome (the behavior of the runoff) by describing the physical processes that occur between, and ultimately affect, the rainfall-runoff relationship. The processes occurring within the control volume of the watershed can include evapotranspiration, infiltration, and storage, each of which has a physical consequence on the amount of runoff that will occur from a given precipitation event. Subsequently, engineering decisions that affect these physical processes are investigated closely for their influences on the rainfall-runoff relationship. This strain of inquiry forms the basis for the preponderance of scientific studies that utilize SWMMs modeling capacity (e.g. Cipolla et al., 2016; Rosa, Clausen, & Dietz, 2015; Tobio, Maniquiz-Redillas, & Kim, 2015).

SWMMs strength, the capacity to use physical processes to describe the behavior of water in a catchment, is also a point of weakness, as it requires an increased number of parameters that need to be assigned value accurately in order to effectively match the observed runoff behavior. Compounding this, many of these parameters, such as Manning's roughness or infiltration coefficients, vary by orders of magnitude and are difficult to estimate precisely at scales larger than a

laboratory setting. Rigorous scientific research has yielded ranges for these and other parameters, many of which can be found in the SWMM user's guide (Rossman et al., 2010) or other such handbooks (Maidment, 1993). For the majority of scientific research, these ranges do not offer the precision necessary to make useful predictions, so some level of calibration is needed (e.g. Guan et al., 2015b; C. Li, Liu, Hu, Gong, & Xu, 2016; J. Li, Li, & Li, 2016).

1.3.2 "Calibration" Broken Down

Rossman describes calibration as more than just one step (Rossman et al., 2010). The common term "calibration" is broken down into sensitivity analysis, actual calibration, and uncertainty analysis. However, in the literature on calibration in hydrologic modeling conducting all three sub-steps of calibration is extremely uncommon; some do the first two steps of sensitivity analysis and calibration in an automated fashion (Krebs, Kuoppamäki, Kokkonen, & Koivusalo, 2016), while others do these steps, but some portion of the analysis is done manually (Knighton, White, Lennon, & Rajan, 2014; C. Li et al., 2016). These three sub-steps create a natural checklist by which studies can be compared.

- ☐ Sensitivity Analysis
- ☐ Calibration
- ☐ Uncertainty Analysis

Studies that demonstrate the presence of more than one of these steps are considered to be more sophisticated than those that did one or none, and studies that show that the steps were done with the assistance of automated algorithms

are considered to be more sophisticated than those that conducted the analysis manually.

Sensitivity analysis describes the process of determining which parameters affect the behavior of the model the most, so that the "insensitive" parameters can be justifiably neglected. When sensitivity analysis precedes calibration, it can have the effect of speeding up the calibration process, due to a decreased number of degrees of freedom, or dimensionality of the problem. A number of researchers take advantage of this speed-up, especially when conducting calibration manually (Guan et al., 2015a, 2015b; Rosa et al., 2015). Sensitivity-based approaches to calibration have also been formally integrated into SWMM, although not in the EPA-approved downloadable model. The proprietary package PCSWMM makes use of the Sensitivity-based Radio Tuning Calibration (SRTC) tool (CHI Water, 2018). The SRTC tool works by evaluating the SWMM model at the edges of the feasible parameter space to develop a number of extreme base cases. The behavior of the SWMM model given intermediate values within this parameter space can be linearly interpolated between the base cases (Yim, Aing, Men, & Sovann, 2016).

Calibration itself is the process of comparing model performance to some observed data set and making adjustments to the model parameters such that the abstract model more closely matches the real-world observations. Calibration does not strictly require sensitivity analysis as a prerequisite and some researchers do not conduct one (Barco, Wong, & Stenstrom, 2008; Masseroni & Cislaghi, 2016). The penalty of not pre-processing a calibration problem with a sensitivity analysis is an increase in computational expense, as well as the logical potential for algorithms to struggle to converge to optimal values for parameters that do not affect the behavior of the model. Barco describes several algorithms that have gained

popularity in the calibration context (Barco et al., 2008) – Artificial Neural Networks (ANN), the complex method (Box, 1965), and the NSGA-II algorithm (Deb et al., 2002). Barco, however, did not chose one of the aforementioned methods, and opted for an aggregated approach to multi-objective function optimization (Barco et al., 2008).

The third sub-step, uncertainty analysis, is best described by contrasting it against the validation phase. Validation is the process of determining the validity of a calibrated model by comparing its performance against some data that was not used for calibration. In this way, the validation phase provides a measure of the uncertainty of the model as a whole. Uncertainty analysis, on the other hand, is the process of determining the uncertainty in the solutions of each calibrated parameter (Knighton, Lennon, Bastidas, & White, 2016). Generalized likelihood uncertainty estimations (GLUE), Bayesian likelihood functions, and distributions derived from Markov-chain Monte Carlo simulation are optional strategies for analyzing the uncertainty in the individual parameter solutions (Knighton et al., 2016; Muleta, McMillan, Amenu, & Burian, 2013; Sun, Hong, & Hall, 2013; Zhang, Li, & Dai, 2015).

For the present work, the NSGA-II algorithm, was chosen to be incorporated as the optimization engine for SWMMCALPY. The NSGA-II algorithm as applied to SWMM is a fairly well-investigated topic (e.g. Herrera, Heathcote, James, & Bradford, 2006; Krebs, Kokkonen, Valtanen, Koivusalo, & Setälä, 2013; Krebs, Kokkonen, Valtanen, Setälä, & Koivusalo, 2014; Krebs et al., 2016; Shan & Wang, 2005; Shinma & Reis, 2011, 2014).

Methods for conducting sensitivity analysis and uncertainty analysis are discussed in the literature review but ultimately omitted from the initial development

of SWMMCALPY. For this thesis, SWMMCALPY was constructed to converge to a solution, and so only required the calibration sub-step. Considerations were made to allow for sensitivity and uncertainty analysis steps to be retroactively implemented. Python libraries exist to allow for insertion of a GLUE method for uncertainty analysis to the end of SWMMCALPYs workflow (Python.org, 2018).

1.4 Introduction to Python & SWMM

Initially developed in 1989, Python has grown to be one of the world's most popular coding languages (Misirlakis, 2017). Python versions 2.0 and newer have all been in the open-source domain, differentiating python from coding tools like MATLAB, which is fairly well supervised by MathWorks (Moler, 2004). Python's versions 2.7 and 3.x are both commonly used permutations of the base python language (with very limited inter-version compatibility) making python's community unusually fragmented (Wood, 2015). Python does grant myriad advantages, however. Python is an interpreted, object-oriented, high-level language with community support and excellent documentation (Python.org, 2018), all of which contributed to the development of SWMMCALPY. Python's ability to consolidate data from disparate sources via wrappers makes it an acceptable choice for dealing with the input files SWMM uses to contain model information.

Python's usefulness dealing with the SWMM workflow has already been investigated. The Open Water Analytics group developed the PySWMM project as the foundation for their OWA-SWMM toolkit API. PySWMM is a low-level API that uses abstractions from the C-code underlying SWMM itself to eschew the broken interactions workflow that has historically characterized SWMM optimization. A broken interactions workflow designates that additional input files

are created to contain the calibrated parameter values, rather than adjust the original input file. PySWMM accesses the SWMM input and report files directly to get parameter values or interact with the model results in a novel, transparent way (McDonnell, 2018a, 2018b). PySWMM was leveraged in the development of SWMMCALPY.

The SWMM-UI is also built using python (Contributors, 2018). This is different from the EPA SWMM version 5.1 GUI. Two influential open-source groups are using primarily python tools for their future SWMM development, so it is clear that python is an acceptable choice for the production of SWMM support tools. The powerful and state-of-the-art python methods employed is one of the primary strengths of SWMMCALPY as a calibration routine.

Open-source tools supporting SWMM have been developed in other languages. RSWMM was designed by researchers at Virginia Tech to automatically calibrate SWMM using the statistical language, R (Alamdari et al., 2017). RSWMM also uses the NSGA-II algorithm and multiple objective functions. RSWMM helps fill the space for open-source autocalibration routines for SWMM, but differs in several ways. Both the primary language used and the objective functions employed vary between RSWMM and SWMMCALPY. RSWMM also does not consider uncorrelated subcatchment parameters the way SWMMCALPY does.

Another software, MatSWMM, is a MATLAB-based toolbox build to aid engineers in real-time investigations of drainage systems (Riaño-Briceño, Barreiro-Gomez, Ramirez-Jaime, Quijano, & Ocampo-Martinez, 2016). The access MatSWMM grants to the user in terms of supervising to the traditional SWMM workflow puts it in a similar category as PySWMM. Both PySWMM and MatSWMM have the objective of further fracturing the hydrologic black box and enhancing SWMM simu-

lation transparency. A distinguishing feature between SWMMCALPY, RSWMM, and MatSWMM, is that RSWMM and MatSWMM were designed for research purposes and are loosely maintained (Alamdari et al., 2017; Riaño-Briceño et al., 2016). SWMMCALPY is intended to be used for design purposes.

1.5 SWMMCALPY Objectives

SWMMCALPYs fully-automated routine eliminates the need for direct engineer interference between the algorithm inputs and the returned solution. Running SWMMCALPY only requires the initialization work of specifying a SWMM input file to be calibrated, a root location of observed data within the model system, and selecting weights for the objective functions. The routine then returns a calibrated SWMM input file to the user, along with performance statistics.

To achieve this holistic goal in a way that is the most useful to the end user, SWMMCALPY also has three sub-objectives, or strategies. These sub-objectives also behave as checkpoints in the SWMMCALPY workflow, each solving a portion of the overall problem and passing a minimalist argument to the next sub-objective. The sub-objectives are:

1. to define a variable calibration scope
2. to determine the Pareto front for multiple objective functions
3. to isolate the optimal solution as informed by user-defined weights

Figure 1.1 illustrates the user inputs necessary for SWMMCALPY. It also broadly depicts SWMMCALPYs workflow segmentation and terminology associated with that operational section.

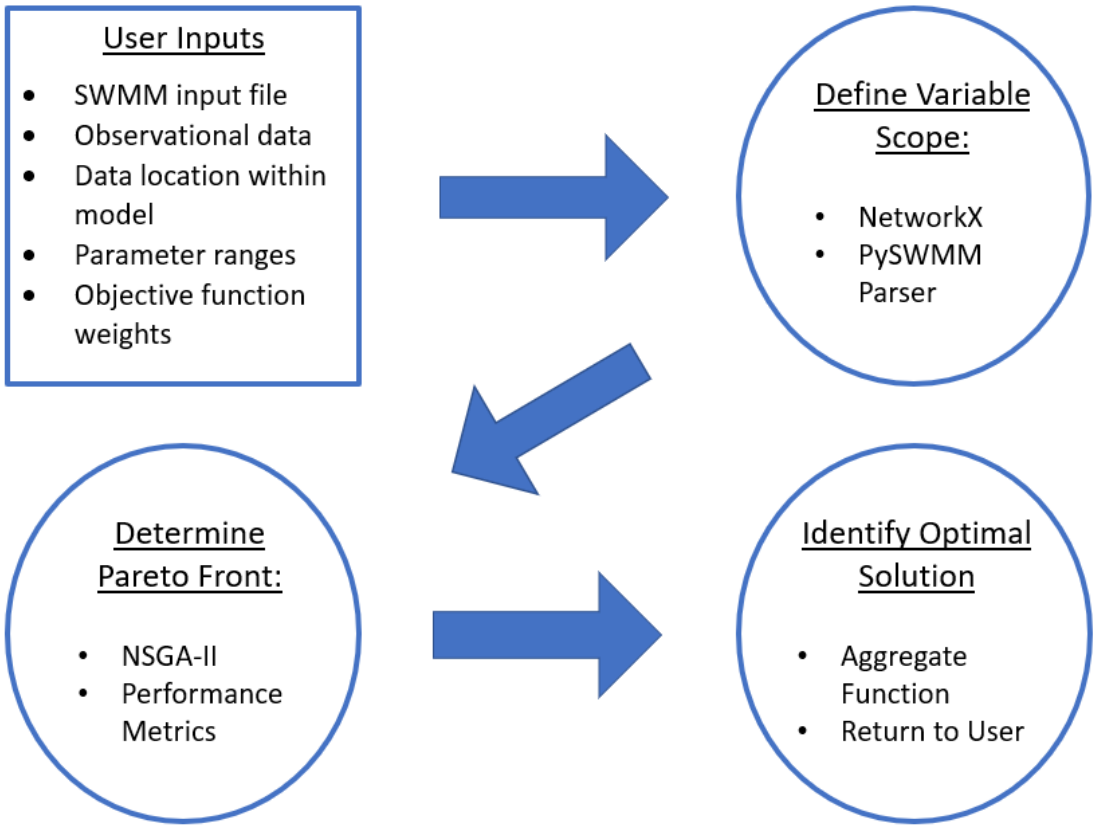


Figure 1.1: Broad description of SWMMCALPYs Workflow from User Inputs to the Identification of an Optimized Solution.

1.5.1 Define Variable Calibration Scope

One issue SWMMCALPY seeks to address is the issue of generality in terms of observed data availability in real world systems. Improving the accuracy of model parameters requires observed data for calibration. These observed data could be collected at the outfall of the system, which would suggest that the entire catchment drains through the observed location and therefore is calibratable as a single unit. Observed data could also be collected somewhere else in the system, indicating that only a subset of the subcatchments that make up a SWMM model are calibratable with the given information. SWMMCALPY has the capacity to isolate the scope of a calibration problem given only the location of the observed data.

Figure 1.2 and Table 1.1 are included to illustrate how different root locations within a SWMM system result in different clipped subsystems. Figure 1.2 and Table 1.1 are repeated in §2.3 where they are explained in greater detail.

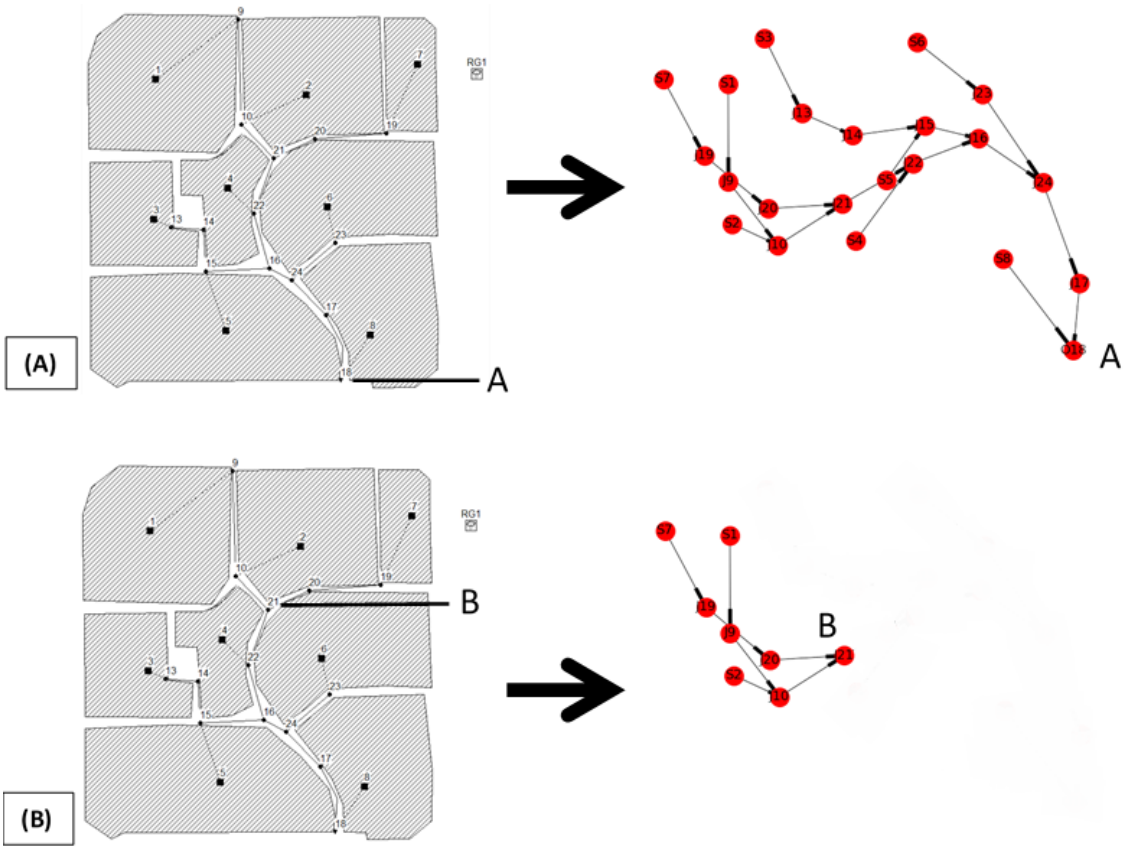


Figure 1.2: Example1 transformed into directed graph object with corresponding locations of A) the outlet and B) an upstream junction being passed to SWMMCALPY for the list of contributing subcatchments to be extracted.

Table 1.1: SWMMCALPY intermediate output, the list of contributing subcatchments for cases shown in Figure 6A and 6B.

(A)	['S1', 'S2', 'S3', 'S4', 'S5', 'S6', 'S7', 'S8']
(B)	['S1', 'S2', 'S7']

An effect of allowing a variable scope for calibration is that each subcatchment is allowed j degrees of freedom, where j is the number of parameters being calibrated. Variable scope calibration implies that SWMMCALPY is not limited to cases where the parameters for each subcatchment are strictly correlated. This as-

sumption is not applied to some calibration strategies, such as the SRTC tool used by PCSWMM, which assigns one value of each parameter to all subcatchments in the system (CHI Water, 2018). Disadvantages of SWMMCALPY's generalization of calibrated parameter values include increased computational complexity.

1.5.2 Determine Pareto Front for Multiple Objective Functions

Another engineering consideration SWMMCALPY addresses is the usefulness of multiple objective functions. A storm water management engineer might, for a given SWMM model purpose, be interested in various aspects of model behavior. For that reason, SWMMCALPY aims to minimize a cadre of objective functions including peak flow error, volume of flow error, a modified Nash-Sutcliffe Efficiency (NSE) index, and a distance function that insists the algorithm prefer solutions near the user's original parameter estimates. These objective functions used in the initial development of SWMMCALPY are written into the workflow as replaceable modules. Model results can be evaluated using different criteria merely by providing new constraint equations and appropriate weighting within SWMMCALPY.

The NSGA-II algorithm employed by SWMMCALPY converges to the Pareto front for multiple object functions (Deb et al., 2002). This Pareto front is defined as the surface in n-dimensional space (in the context of SWMMCALPY, 4-dimensions) along which it is impossible to improve the value of one objective function without worsening the value of at least one other. A graphical representation of the Pareto front in 2-dimensions is shown in Figure 1.3.

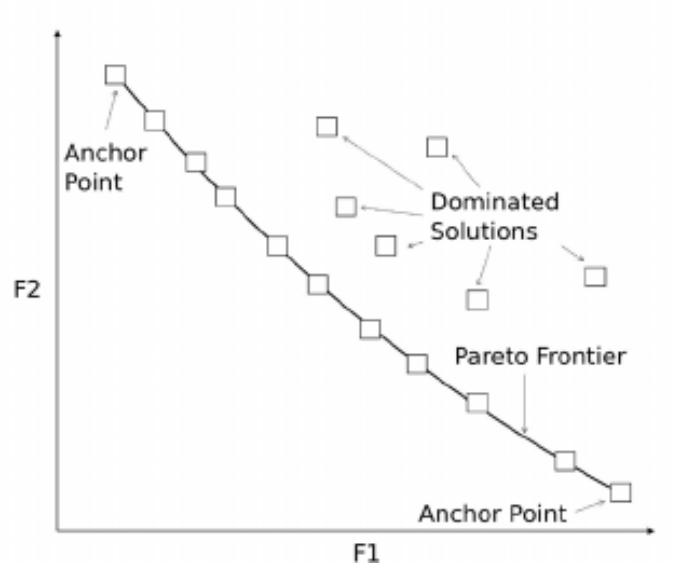


Figure 1.3: 2-dimsional depiction of the Pareto front for minimizing objective functions F1 and F2. From Oliveira and Van Noije (2012).

In the jargon of the NSGA-II method, "Anchor Points" are the global minima for either F1 or F2, irrespective of the value of the other and "Dominated Solutions" are values such that it is possible for both F1 and F2 to be improved (i.e. that solution cannot exist on the Pareto front) The Pareto front, or "Pareto frontier" in Figure 1.3, represents essentially the optimal cases for combining F1 and F2. In Figure 1.3, the Pareto front is shown with a convex shape, but the shape (and location) of the Pareto front varies with the system and the objective functions used. Without an independent, additional selection criteria, it is impossible to compare the goodness of points along the Pareto front. The difference between this "selection criteria" and another "objective function" is explained in the following section, §1.5.3.

Once progress towards the Pareto front ceases, the NSGA-II halts its iterative process. In the SWMMCALPY workflow, the NSGA-II is solving a con-

strained optimization problem. Each parameter being calibrated will only be allowed to vary within an engineering feasible range, thus creating what will be referred to as the "feasible parameter space" (Knighton et al., 2016). The NSGA-IIs Pareto front determination occurs in what Shan and Wang (Shan & Wang, 2005) coined the "feasible performance space".

1.5.3 Isolate Optimal Solution with User-Defined Weights

The selection criteria for comparing Pareto-front solutions makes use of user-defined weights on the various objective functions. These weights differentiate one engineering problem from the next. For example, a catchment being studied for flooding might have different function weights than the same catchment being studied for low-conditions. The object function weights are used by SWMMCALPY to inform the routine on its selection of a final solution, after the NSGA-II algorithm has stopped iterating. By relating the weights to one another, an optimal angle can be defined. The solution in the final generation of the NSGA-II algorithm closest to this angle corresponds to the set of parameters that is returned.

The SWMMCALPY routine prompted with a base SWMM input file, observational data + location, and objective function weights will return a new SWMM input file with calibrated parameters. Associated performance statistics accompany the calibrated SWMM input file.

Chapter 2: Methodology

This chapter presents the methodology of SWMMCALPY. Details on the software used are presented first. An introduction to the simple SWMM model used for verification of intermediate steps is also provided. Furthermore, each of the sub-objectives that collectively comprise the SWMMCALPY workflow is explored in detail.

2.1 Software Used

SWMM 5.1.012 was downloaded from the official EPA website (EPA, 2014).

Python version 3.6.5 was downloaded from python.org (Python.org, 2018). A 32-bit instance of this python version was installed to comply with the requirements of the python libraries used. Details on the specific libraries used are presented in later sections.

Python scripting was done using the PyCharm Community Integrated Development Environment (IDE) obtained from jetbrains.com (Jetbrains, 2018).

2.2 Example file for SWMMCALPY Development

Throughout the development of SWMMCALPY routine, a simple SWMM system was used to verify the functionality of intermittent steps. The simple SWMM system was obtained from the SWMM5 tutorial and shall be referred to as "Example1.inp". Example1.inp – as read by the EPA SWMM 5.1 GUI – is depicted in Figure 2.1.

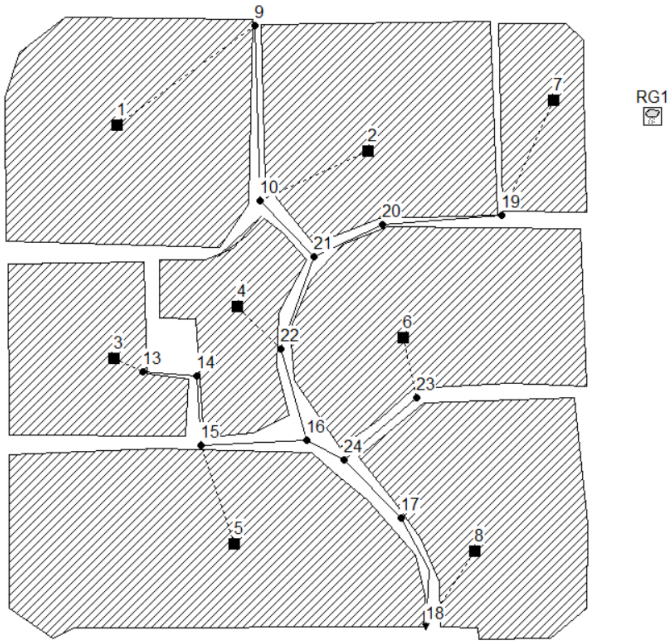


Figure 2.1: EPA acrshortSWMM 5.1 interpretation of the Example1.inp file.

Example1.inp is characterized by 8 subcatchments labeled 1-8 and 14 junctions labeled 9-10 and 13-24. From the Example1.inp file, shown in Table 2.1, each subcatchment an area of 10 acres. Forcing time series used in the Example1 model are synthetic.

Table 2.1: A portion of the Example1.inp file containing subcatchment properties.

[SUBCATCHMENTS]

;;Name	Rain Gage	Outlet	Area	%Imperv	Width	%Slope	CurbLen
1	RG1	9	10	50	500	0.01	0
2	RG1	10	10	50	500	0.01	0
3	RG1	13	10	50	500	0.01	0
4	RG1	22	10	50	500	0.01	0
5	RG1	15	10	50	500	0.01	0
6	RG1	23	10	10	500	0.01	0
7	RG1	19	10	10	500	0.01	0
8	RG1	18	10	10	500	0.01	0

[SUBAREAS]

;;Subcatchment	N-Imperv	N-Perv	S-Imperv	S-Perv	PctZero	RouteTo	PctRouted
1	0.001	0.10	0.05	0.05	25	OUTLET	
2	0.001	0.10	0.05	0.05	25	OUTLET	
3	0.001	0.10	0.05	0.05	25	OUTLET	
4	0.001	0.10	0.05	0.05	25	OUTLET	
5	0.001	0.10	0.05	0.05	25	OUTLET	
6	0.001	0.10	0.05	0.05	25	OUTLET	
7	0.001	0.10	0.05	0.05	25	OUTLET	
8	0.001	0.10	0.05	0.05	25	OUTLET	

Table 2.1 shows relevant subcatchment properties (e.g. area, outlet junction name) and the model parameters addressed in SWMMCALPYs calibration exercise. In the initial development of SWMMCALPY only the subcatchment surface parameters (excluding area) were treated as calibratable. Typically, the subcatchment area is either selected intentionally or derived from fairly precise GIS tools (Geosyntec, 2017), so, is not generally a target for calibration. Drainage network parameters (e.g. *CurbLen*, *PctRouted*) and infiltration parameters will be considered in future work with SWMMCALPY.

The subcatchment *Name*, *Rain Gage*, *Outlet*, and *RouteTo* are all selected variables that inform how the model is constructed. They are not calibratable. Thus, there are eight calibratable subcatchment parameters: *%Imperv*, *Width*, *%Slope*, *N-Imperv*, *N-Perv*, *S-Imperv*, *S-Perv*, and *PctZero* to form the basis of

the calibration problem to be solved by SWMMCALPY. Table 2.2 describes the parameters.

Table 2.2: Input file variable names and descriptions. From Rossman et al. (2010).

Width	Characteristic width of the overland flow path for sheet flow runoff (feet or meters). An initial estimate of the characteristic width is given by the subcatchment area divided by the average maximum overland flow length. The maximum overland flow length is the length of the flow path from the inlet to the furthest drainage point of the subcatchment. Maximum lengths from several different possible flow paths should be averaged. These paths should reflect slow flow, such as over pervious surfaces, more than rapid flow over pavement, for example. Adjustments should be made to the width parameter to produce good fits to measured runoff hydrographs.
% Slope	Average percent slope of the subcatchment.
% Imperv	Percent of land area which is impervious.
N-Imperv	Manning's n for overland flow over the impervious portion of the subcatchment (see Chapter 24, section 24.6 for typical values).
N-Perv	Manning's n for overland flow over the pervious portion of the subcatchment (see Chapter 24, section 24.6 for typical values).
Dstore-Imperv	Depth of depression storage on the impervious portion of the subcatchment (inches or millimeters) (Chapter 24, section 24.5).
Dstore-Perv	Depth of depression storage on the pervious portion of the subcatchment (inches or millimeters) (Chapter 24, section 24.5 for typical values).
% Zero-Imperv	Percent of the impervious area with no depression storage.

For the remainder of Chapter 2 the methodology presented uses the Example1.inp simple SWMM model. Subsequently, in Chapter 3, a more compelling test case is introduced to provide evidence for the robustness and effectiveness of SWMMCALPY on a more complex SWMM model.

2.3 Define Variable Calibration Scope

The logic behind defining a variable calibration scope is first enumerated in Chapter 1 and is restated here. If a storm-driven hydrograph is obtained through observation at a given location, then that hydrograph contains information only about SWMM subcatchments whose runoff flowed through that location. It does

not contain any information about subcatchments whose runoff flowed elsewhere. For this reason, it is important for SWMMCALPY to produce a clipped SWMM subsystem of only contributing areas to the observational data location.

An effective means of producing this clipped subsystem for any SWMM model was found by making use of the NetworkX library in python. The NetworkX library package operates by converting any system into a "graph object" composed of a web of "nodes" and "edges". Various operations can then be applied to the graph object that were not otherwise possible for the general system (Johnson & Stinchcombe, 2007). Converting a SWMM model into a NetworkX graph object is fairly simple as the corollaries from various SWMM objects to NetworkX's nodes and edges are strong. SWMMs subcatchments, junctions, and outfalls were transformed into nodes. Outlets, conduits, weirs, pumps, and the like were transformed into edges. A constraining assumption that water will only flow in one direction within the system (i.e. no backwater effects permitted) allowed the entire SWMM model to be cast as a "directed" graph object, enabling further functionality.

Unfortunately, the information that informs the construction of the directed graph object is carried in a variety of places in the SWMM input file. To extract the NetworkX-critical information, an SWMM input file parser is required. SWMMCALPY invokes the PySWMM wrapper for this purpose.

PySWMM is a python wrapper for the SWMM workflow that contains functions for accessing aspects of the SWMM model blueprint (McDonnell, 2018b). Once the input file under consideration is passed to PySWMMs "Simulation" module, relationships between SWMMs objects can be accessed and passed to NetworkX for transformation into the final directed graph object. Inconveniently,

PySWMM has its own vernacular to describe SWMM objects, which partially conflicts with NetworkX's parlance. In PySWMM, "Nodes" correspond to SWMMs junctions and outfalls. "Subcatchments" comprise a PySWMM object all to themselves. "Links" in PySWMM translate directly to NetworkX "edges", apart from the omission of subcatchment outlets, which do not have an associated PySWMM object. To circumvent the lack of a PySWMM-NetworkX translation for subcatchment outlets, a manual parser was built to supplement PySWMM and adds these connections as NetworkX edges directly.

We can think of the process as PySWMM reading the SWMM file and passing it to NetworkX to create a directed graph object for use by SWMMCALPY. A commented python code file, "DelineateNetwork.py", is provided in §C.1 with the script used to execute this transformation. "DelineateNetwork.py" exists as one of the python scripts comprising the SWMMCALPY routine. Figure 2.2 shows the result of this transformation on the Example1 model.

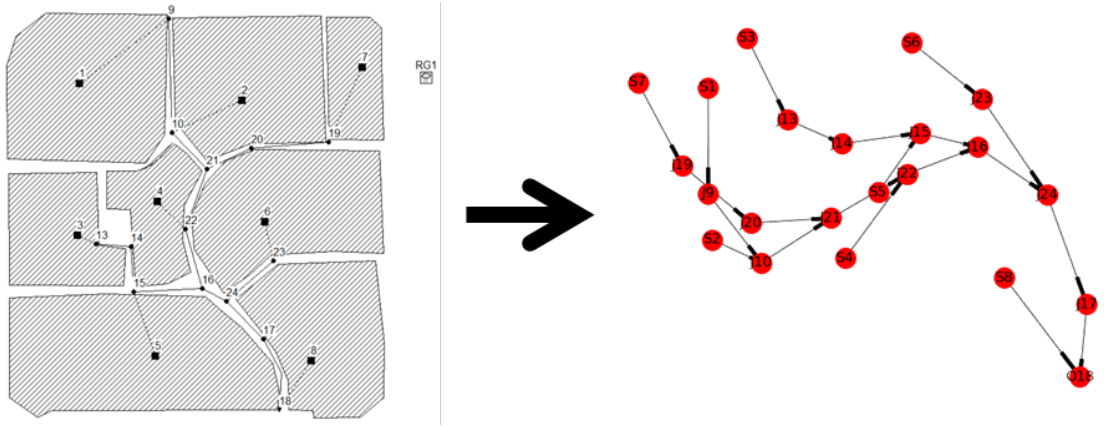


Figure 2.2: EPA SWMM 5.1 interpretation of Example1.inp transformed into a directed graph object. The red circles represent nodes and the dark arrows are the edges preserving direction.

The SWMM system becomes "traversable" in NetworkX when processed

into a directed graph object. This means that the relationships between distant nodes can be investigated much more easily than with original, unprocessed input file. NetworkX contains a "predecessor" function for directed graph objects that returns the upstream neighboring nodes to any root node. Recursive calls to this predecessor function on each upstream neighbor identifies the entire network. Thus, when the node corresponding to the location of the observational data is passed to the recursive call, the entire contributing subsystem is returned. This method is preferred over more efficient tree search algorithms because SWMM-CALPY is not constrained to systems with tree branch-like shapes.

This method is flexible enough to handle cross-cutting connections or redundancies that are typical of real-world SWMM models (Geosyntec, 2017) due to the NetworkX property that a graph object will ignore any recurring addition. Figure 2.3 and Table 2.3 were originally presented as Figure 1.2 and Table 1.1 in §1.5.1, and are re-shown here to illustrate the output of the variable calibration scope to different root locations within the Example1.inp system.

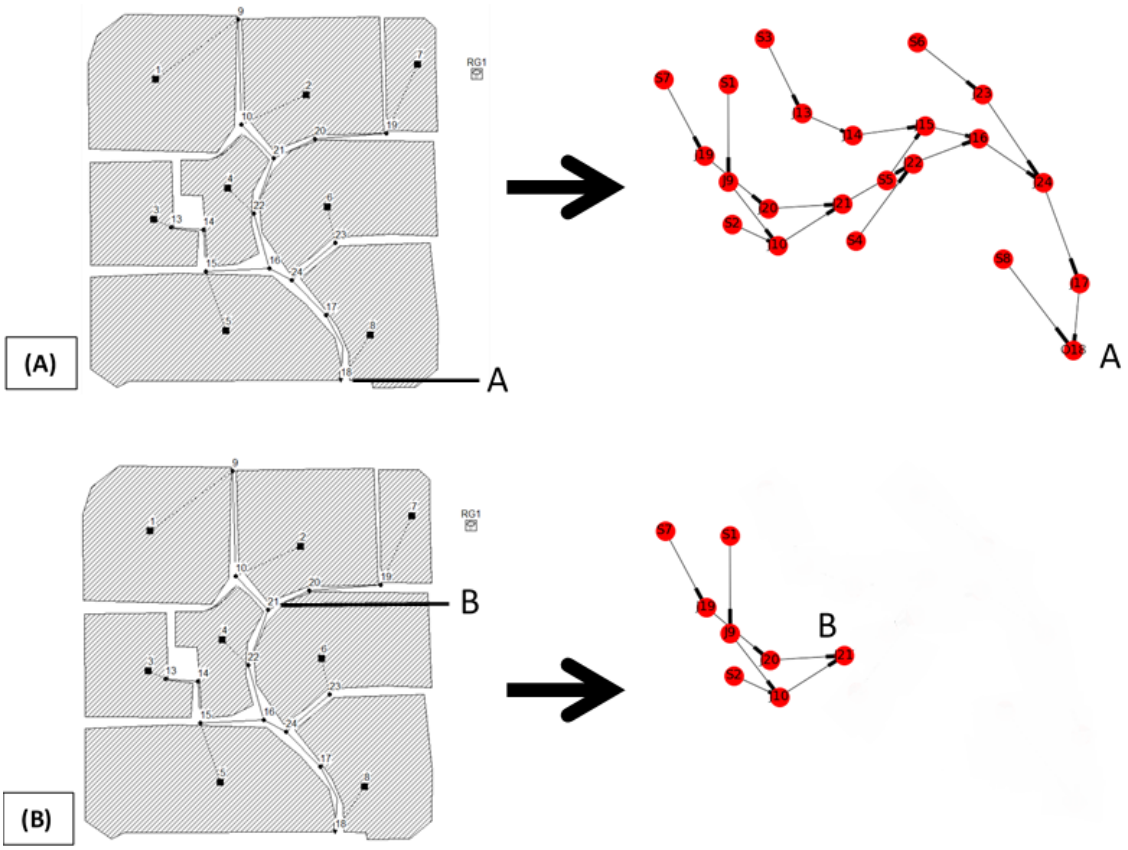


Figure 2.3: Example1 transformed into directed graph object with corresponding locations of A) the outlet and B) an upstream junction being passed to SWMM-CALPY for the list of contributing subcatchments to be extracted (Again).

Table 2.3: SWMMCALPY intermediate output, the list of contributing subcatchments for cases shown in Figure 6A and 6B (Again).

(A)	`['S1', 'S2', 'S3', 'S4', 'S5', 'S6', 'S7', 'S8']`
(B)	`['S1', 'S2', 'S7']`

From a graph object comprising a clipped version of the larger system's nodes and edges, it is easy to extract only the subcatchments included by comparing the subsystems nodes to the values held within PySWMM's "Subcatchments" object. The end result is the comprehensive list of subcatchments whose runoff

flows through the observational data location. Figure 2.3 and associated Table 2.3 help clarify the above description by providing output for two different cases of root locations in the system from Figure 2.2 being passed to SWMMCALPY.

The information presented in Figure 2.3 and Table 2.3 ultimately represents the scope of the calibration problem should the observational data exist at either points A or B. Only the subcatchments listed in Table 2.3 will be subject to the NSGA-II manipulation in the subsequent parts of the SWMMCALPY workflow. Note that this approach requires that the observational data must be located at a junction in the SWMM model. In general, observations are made at manholes or other discrete features in a drainage network (e.g. weirs), so this restriction should not be burdensome. In the unusual case where data are available along some channel length, SWMMCALPY requires the drainage system to be defined so that a junction is included at the observed data location. An interesting calibration challenge is posed by systems with multiple observed data sets at different locations that have overlapping calibration scopes. This challenge is discussed §5.2.

2.4 Determine Pareto Front for Multiple Objective Functions

The hydrologic rationale for multiple objective functions, as well as concept of a Pareto front for these functions, was introduced in Chapter 1 and is revisited here in greater detail. In general, calibration cannot make a model hydrograph exactly match observations. The *art* of calibration then lies in deciding what constitutes a "best" match, which can be quantified by some combination of measures or "objective functions". If a single objective function is used, then calibration becomes a matter of minimizing the objective function (or maximizing, depending on its formulation). Where multiple objective functions are of interest they can

be combined to create a "Pareto" front that is the locus of all the potentially optimum calibration solutions. The selection of the best match then depends on the weighting provided to the different objectives. The goal of automated calibration is to force the decisions on weighting, objective limits, and the parameter limits to be made prior to calibration rather than as *ad hoc* interventions before or after calibration when the results are disagreeable.

A key challenge is that each objective function ideally should be independent, otherwise the weighting will double-count the dependent effects. In practice, there will typically be some unavoidable overlap between different objective functions.

Eleven common objective functions for hydrologic modeling have been previously analyzed (Shinma & Reis, 2011). They determined that metrics of absolute peak flow error, absolute volume error, and relative squared error most nearly satisfied the independence condition. Minimizing these specific functions captures the various aspects of a hydrograph: the maximum flow rate, the total amount of water flowing through the system, and the general shape of the hydrograph as a time series. These objective functions were considered for use in the NSGA-II algorithm within SWMMCALPY, but are conceptually difficult to combine with a simple weighting approach. As a modified approach, the peak flow error and absolute volume error were normalized as the NPE and NVE, respectively, which provides each in a continuous and bounded range of [0,1] where zero is optimum. A modified Nash-Sutcliffe Efficiencey (NSE_m) and a normalized Euclidean distance (NED) were also included as objective functions. Figure 2.4 contains a schematic representation of the hydrological value added by each objective function.

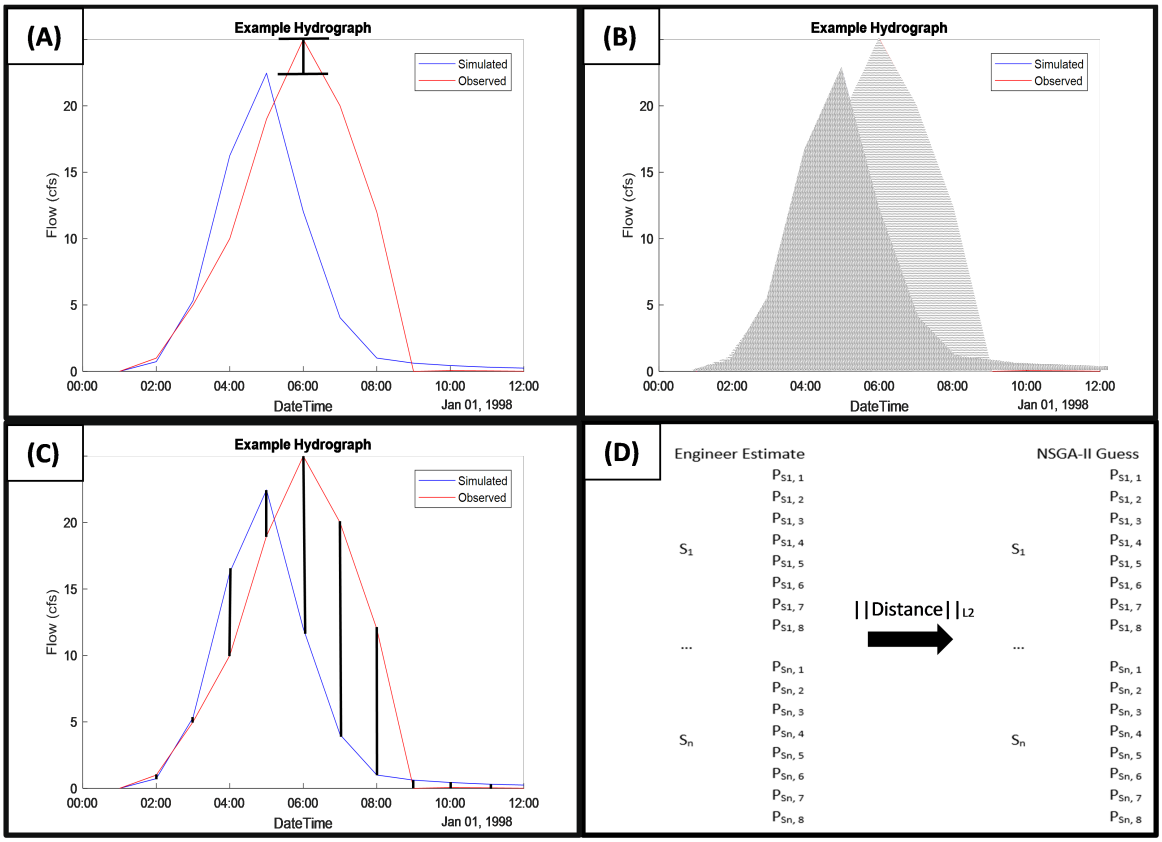


Figure 2.4: Schematic representation of the hydrologic context behind each of the four objective functions used in the initial development of SWMMCALPY. A) NPE - peak error; B) NVE - volume error; C) NSE_m - measure of curve fit at each time instance; and D) NED - distance between parameter sets in the feasible parameter space.

The mathematical formulations for NPE and NVE are specifically:

$$NPE = \frac{|max(Q_{(obs)i}) - max(Q_{(sim)i})|}{max(Q_{(obs)i}) + max(Q_{(sim)i})} \tag{2.1}$$

$$NVE = \frac{|V_{obs} - V_{sim}|}{V_{obs} + V_{sim}} \tag{2.2}$$

where Q is the instantaneous flow rate, V is the volume, subscripts $(obs)i$

and $(sim)i$ indicate the i^{th} element of the observation or simulation time series, respectively. A trapezoidal approximation for volume - assuming a constant time delta for both timeseries - is used in Eq. 2.2. A general trapezoidal approximation for volume is shown in Eq. 2.3.

$$V = \sum_{i=1}^{N-1} \frac{(f(t)_{i+1} + f(t)_i) * (t_{i+1} - t_i)}{2} \tag{2.3}$$

In Eq. 2.3 $f(t)$ is some function of t and N is the number of elements in the series $f(t)$.

The relative squared error discussed by (Shinma & Reis, 2011) was replaced with a observation-normalized root-mean-square error (RMSE), which can be thought of as a modified Nash-Sutcliffe Efficiency. The NSE, used frequently in hydrologic analysis (e.g. De Paola, Giugni, & Pugliese, 2016; Khu & Madsen, 2005; Madsen, 2000), is given formally as by Nash and Sutcliff (Nash & Sutcliffe, 1970):

$$NSE = 1 - \frac{\sum_{i=1}^{N}(Q_{(obs)i)} - Q_{(sim)i))^2}}{\sum_{i=1}^{N}(Q_{(obs)i)} - \overline{Q_{obs}})^2} \tag{2.4}$$

where $\overline{Q_{obs}}$ is the mean of the observed time series, computed as:

$$\overline{Q_{obs}} = \frac{1}{N} \sum_{i=1}^{N} Q_{(obs)i} \tag{2.5}$$

We define a modified NSE as

$$NSEm \equiv 1 - NSE \tag{2.6}$$

If follows that

$$NSE_m = \frac{\sum_{i=1}^{N}(Q_{(obs)i)} - Q_{(sim)i)})^2}{\sum_{i=1}^{N}(Q_{(obs)i)} - \overline{Q_{obs}})^2} \tag{2.7}$$

Although the NSE_m uses the same quantifiers as the NSE, the former is scaled over a range of [0, $+\infty$) with zero being optimum (minimization objective function), whereas the later is over the range of (-∞, 1] with unity being optimum (maximization objective function). Thus, our NSE_m is readily combined with NPE and NVE in a combined minimization objective.

As a metric, the NSE index is essentially the ratio of the predictive power of a model to capture the behavior of a timeseries over the predictive power of the average of that timeseries alone. So, for the NSE_m to have a value great than 1, the model would have to perform worse than simply the average of the observed data; or, in other words, terribly. Thus, there is no reason to further constrain the upper boundary of NSE_m to match those of the NPE and NVE. Consistent objective function bounding enables a more transparent comparison between the values of the objective functions. Further details on the logical derivations of Eq. 2.1, Eq. 2.2, and Eq. 2.4 are provided in Appendix D.

Ideally, an automatic calibration routine should preserve existing knowledge of the system and likely conditions. PCSWMMs SRTC tool achieves this explicitly by requiring user-defined constraints on the allowable change in parameters. This establishes upper and lower bounds for the sensitivity analysis to proceed; after which any solution within that feasible parameter space is considered equally valid (CHI Water, 2018).

SWMMCALPY asserts the confidence constraint on calibration slightly differently than PCSWMMs SRTC tool. At the outset, a default feasible parameter space is provided as a general constraint to the optimization problem, with value ranges determined from Chow (1959). In this way, an absolute parameter range constraint is still imposed. However, as the default feasible parameter space was constructed to apply to a wide range of SWMM applications, the value ranges are likely too liberal for a given specific application. Therefore, a fourth objective function was included into the multi-objective function operability of the NSGA-II algorithm within SWMMCALPY. Essentially, this objective function represents the distance, within the feasible parameter space, between the original estimate of parameter values and a parameter set determined by the NSGA-II algorithm. These NSGA-II-derived parameter sets with hereafter be referred to as "guesses". The distance function is referred to as the normalized Euclidean distance NED.

$$NED = \sqrt{\sum_{i=1}^{M} \left(\frac{|O_i - T_i|}{O_i + T_i} \right)^2} \tag{2.8}$$

where O_i and T_i are the i^{th} element of the original and guess parameter sets, respectively. M is the length of the parameter sets. Eq. 2.8 can be interpreted as the L2 norm of the vector composed of the normalized absolute difference between each guess and the original parameter set. Minimizing this distance function would indicate a calibrated solution that is "close" to the originally estimated parameters. Variable confidence on the part of the SWMM user can be controlled by assigning a larger or smaller weight to that objective function.

Three of the four objective functions operate within the feasible performance space, while the fourth is computed within the feasible parameter space. The

hydrological interest in the NPE and NVE, schematically demonstrated in Figure 2.4A and 2.4B, respectively, is fairly straightforward. Inaccurately estimating the peak flow could have consequences for flooding considerations, while errors in volume could affect water balance problems. The NSE_m shown in Figure 2.4C heavily penalizes phase shift, or peak timing errors, which has implications for emergency preparedness, while the NED is a nod to an engineer's confidence in their initial parameter estimates.

Values for the objective functions forms the basis by which guesses can be compared, but many guesses must be generated for a suitable solution to be found. For large dimensional problems, the NSGA-II algorithm is an efficient method by which to generate guesses in the form of potential parameter sets, evaluate their performance and generate new guesses such that subsequent generations better approximate the unique Pareto front for a given problem (Deb et al., 2002; Herrera et al., 2006). Figure 2.5 presents an overview of the NSGA-II workflow and logical basis.

Figure 2.5: Diagram of NSGA-II workflow.

Figure 2.5 details the informational prerequisites to be provided by the user and illustrates the flow of steps taken to properly execute the NSGA-II algorithm. A prosaic description of the overview provided by Figure 2.5 is included here.

Intuitive prerequisites for the NSGA-II algorithm to proceed are a built SWMM model input file with initial values set for each parameter and a defined feasible parameter space to constrain future guesses. Additionally, the algorithm requires some manner of observational (or calibration, as per Figure 2.5) data, to which the model can be compared for calculation of objective functions (including the corresponding model location of that data) and user-defined weights on those objective functions. Another (implied) pre-processing step before the algorithm is able to produce its first generation of guesses is that the provided text files need to be read in so the data is available within the python script. In §2.3, PySWMM was used to parse the network connectivity information from the SWMM simulation module. However, PySWMM was not built as a comprehensive parser and lacked the "getter" functions required to access parameter values under the "[SUBAREAS]" heading. Building off of PySWMMs existing toolbox is possible, but for simplicity a manual parser was created to access the parameters under investigation.

Only the parameter data for subcatchments contributing to the root location are made accessible to the NSGA-II algorithm for manipulation. This screening process is enforced by the list of subcatchment in the clipped subnetwork graph object from the NetworkX analysis in §2.3.

Creating a generation of parameter sets to act as guesses is the first active step in the NSGA-II algorithm. A mutation probability criterion, p_m, is applied repeatedly to a compiled list of contributing subcatchment parameters, creating m

sets of guesses that are uniquely related to the original guess. Genetic algorithms often have a third generational parameter, p_c, representing a "crossover" probability. Crossover probability is the likelihood that two parameter sets in a generation will be combined to produce a guess in the next generation. Crossover probability was neglected in the initial development of SWMMCALPY; each generation is produced purely on the basis of mutation from the previous generation. Table 2.4 details the genetic algorithm variables chosen for the NSGA-II component of SWMMCALPY.

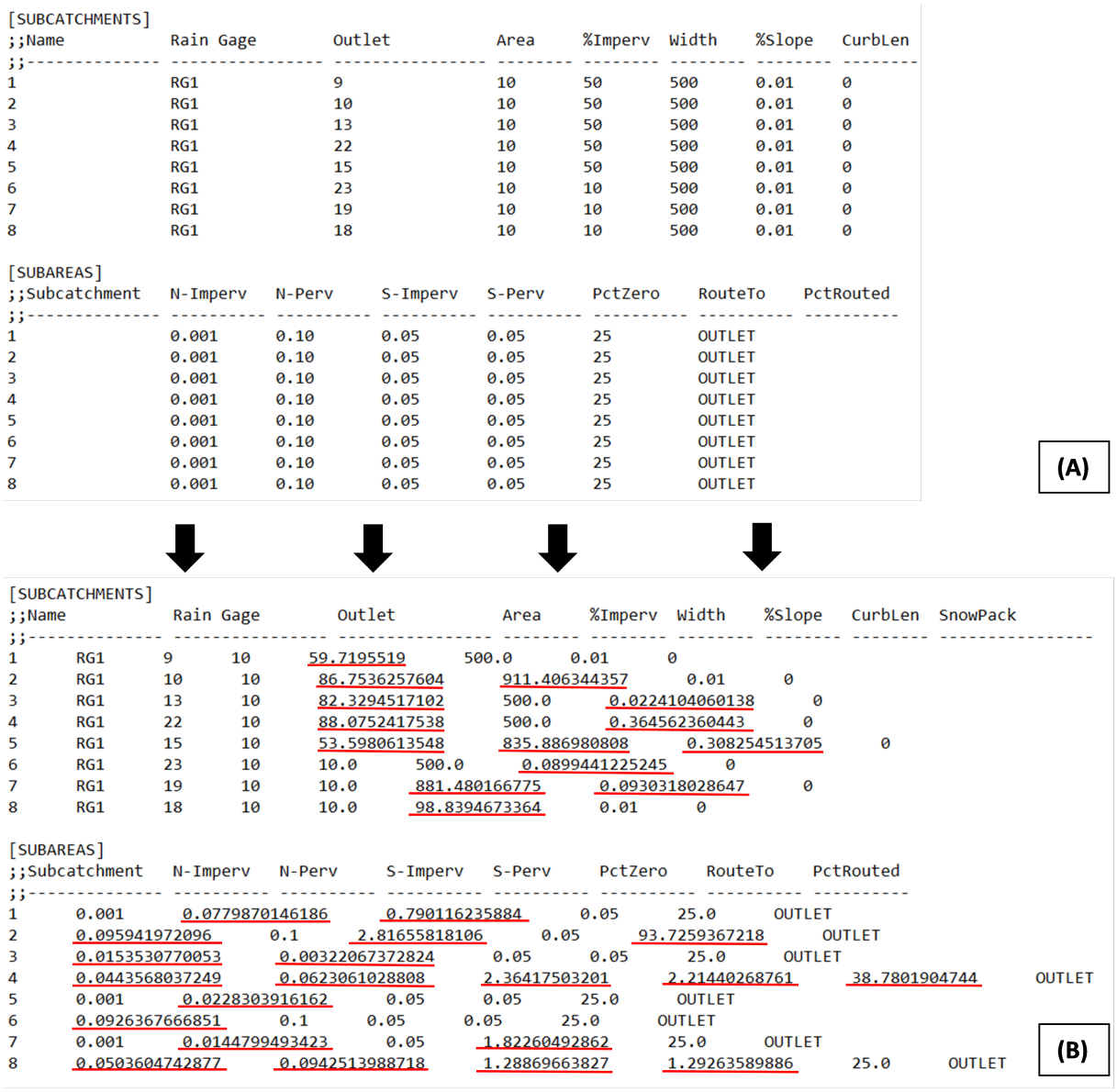
Table 2.4: Genetic parameters used by NSGA-II for producing generations of guesses.

Genetic Algorithm Parameters		
Generation Size	m	100
Mutation Probability	p_m	0.5
Crossover Probability	p_c	0.0

The practical consequence of the values in Table 2.4 is that each parameter in the initial guess had a 50% chance of being mutated. Mutated parameters were randomly given a new value according to a uniform distribution. The upper and lower bounds of the uniform distribution were informed by the default feasible parameter space. The default feasible parameter space in text file form can be found in §B.1. A generation size, m, was chosen to be 100, as that is a typical generation size for NSGA-II (Deb et al., 2002; Shan & Wang, 2005). Others, like Herrera et al. (Herrera et al., 2006), have used $m = 50$. A fairly high value for the mutation probability was chosen because this was the only mechanism causing generational differences. Sensitivity of NSGA-IIs convergence rate to mutation

probability is not known.

The initial generation of guesses, referred to as P, is stored in individual, temporary input files that are identical to the original input file with the exception of the parameters that had been selected for mutation. Figure 2.6 demonstrates an example of the differences between the original SWMM model input file for Example1 and a temporary input file belonging to the first iteration of guesses. Details on the python script responsible for creating each guess and storing it within a temporary input file can be found in the commented python script, "CreateGuesses.py", in §C.2.



[SUBCATCHMENTS]

;;Name	Rain Gage	Outlet	Area	%Imperv	Width	%Slope	CurbLen
1	RG1	9	10	50	500	0.01	0
2	RG1	10	10	50	500	0.01	0
3	RG1	13	10	50	500	0.01	0
4	RG1	22	10	50	500	0.01	0
5	RG1	15	10	50	500	0.01	0
6	RG1	23	10	10	500	0.01	0
7	RG1	19	10	10	500	0.01	0
8	RG1	18	10	10	500	0.01	0

[SUBAREAS]

;;Subcatchment	N-Imperv	N-Perv	S-Imperv	S-Perv	PctZero	RouteTo	PctRouted
1	0.001	0.10	0.05	0.05	25	OUTLET	
2	0.001	0.10	0.05	0.05	25	OUTLET	
3	0.001	0.10	0.05	0.05	25	OUTLET	
4	0.001	0.10	0.05	0.05	25	OUTLET	
5	0.001	0.10	0.05	0.05	25	OUTLET	
6	0.001	0.10	0.05	0.05	25	OUTLET	
7	0.001	0.10	0.05	0.05	25	OUTLET	
8	0.001	0.10	0.05	0.05	25	OUTLET	

(A)

[SUBCATCHMENTS]

;;Name	Rain Gage	Outlet	Area	%Imperv	Width	%Slope	CurbLen	SnowPack
1	RG1	9	10	59.7195519	500.0	0.01	0	
2	RG1	10	10	86.7536257604	911.406344357	0.01	0	
3	RG1	13	10	82.3294517102	500.0	0.0224104060138	0	
4	RG1	22	10	88.0752417538	500.0	0.364562360443	0	
5	RG1	15	10	53.5980613548	835.886980808	0.308254513705	0	
6	RG1	23	10	10.0	500.0	0.0899441225245	0	
7	RG1	19	10	10.0	881.480166775	0.0930318028647	0	
8	RG1	18	10	10.0	98.8394673364	0.01	0	

[SUBAREAS]

;;Subcatchment	N-Imperv	N-Perv	S-Imperv	S-Perv	PctZero	RouteTo	PctRouted	
1	0.001	0.0779870146186	0.790116235884	0.05	25.0	OUTLET		
2	0.095941972096	0.1	2.81655818106	0.05	93.7259367218	OUTLET		
3	0.0153530770053	0.00322067372824	0.05	0.05	25.0	OUTLET		
4	0.0443568037249	0.0623061028808	2.36417503201	2.21440268761	38.7801904744	OUTLET		
5	0.001	0.0228303916162	0.05	0.05	25.0	OUTLET		
6	0.0926367666851	0.1	0.05	0.05	25.0	OUTLET		
7	0.001	0.0144799493423	0.05	1.82260492862	25.0	OUTLET		
8	0.0503604742877	0.0942513988718	1.28869663827	1.29263589886	25.0	OUTLET		

(B)

Figure 2.6: A) Identical to Table 1, portion of Example1.inp that highlights subcatchment parameters; B) Temporary input file, one generation separated from Example1.inp, with the mutated parameters underlined.

To produce Figure 2.6B, Example1.inp input file and the root node of "18" were passed to the generation generator within SWMMCALPY. One of the 100 resulting temporary files was clipped to highlight the "[SUBCATCHMENTS]"

and "[SUBAREAS]" headers. The format asymmetry between the two input files comes from the float precision of the mutated parameters, as well as the fact that each value was tab delimited in Figure 2.6B, as opposed to variably delimited in Figure 2.6A. SWMMs C code compiler uses space delimited tokens, so while the input file in Figure 2.6B is not as aesthetically appealing, it is perfectly legible to SWMM.

Root node "18" corresponds to the outfall of the Example1.inp SWMM model, indicating that all subcatchments in the system were contributing subcatchments. For this particular temporary file, 34 of the 64 mutate-able parameters (8 parameters by 8 contributing subcatchments) were given new values with float precision as determined by a uniform distribution for that parameter's classification. To clarify this point, the "S-Imperv" parameter was sampled from the same uniform distribution for each subcatchment and the distribution is unique to "S-Imperv". Figure 2.6 also serves to reinforce the point that each parameter in the set is subjected to the same mutation probability of 0.5. So, the expected value for the number of mutated parameters in this guess is 32, but the actual number can vary stochastically around that mean.

For the first iteration of NSGA-II, initial guess set, P, is generated. Subsequently, each guess, P_i, is mutated again to form a second generational set, Q. The set $P \bigcup Q$ undergoes SWMM simulation to begin the iterative, genetic process of NSGA-II.

Each temporary input file in the set $P \bigcup Q$ is simulated in SWMM using PySWMM commands and the results are compared to the observational data set provided at the onset of the NSGA-II algorithm execution. PySWMM has the functionality to force the SWMM simulation to report the flow through a node

at a defined timestep. The timestep passed is equal to the time resolution of the observational dataset and currently is only functional for observed data with a consistent timestep. It is worth noting here that, because the Example1 SWMM model being used for the intermediate testing of SWMMCALPYs processes is not a model that exists anywhere on earth, there are no observations. Instead, a synthetic hydrograph timeseries was constructed that had the same basic shape and magnitude as the simulated hydrograph at the Example1 system outfall. Figure 2.7 depicts the synthetic "observed" hydrograph timeseries against the simulated hydrograph for Example1.inp.

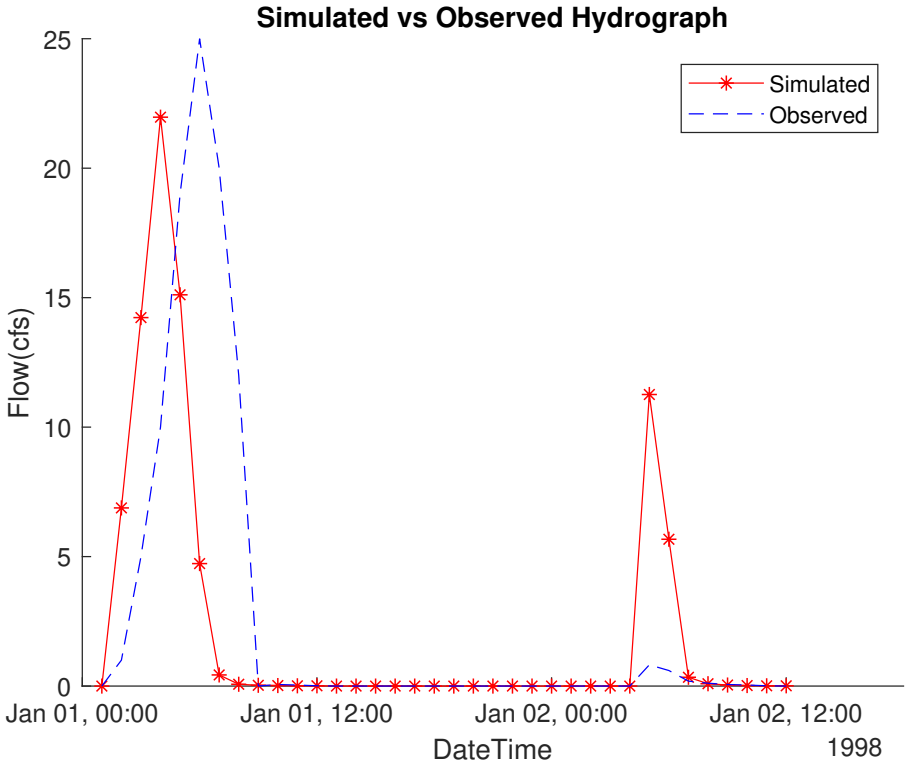


Figure 2.7: Simulated vs "Observed" hydrograph for the outfall of Example1.inp.

Values for the four objective functions being employed can be determined

by comparing these two timeseries for each guess in a generation. Essentially, evaluations of the objective functions transform each guess from a vector of length $8\hat{n}$, where $\hat{n}$ is the number of contributing subcatchments, to a vector of length four that exists in the feasible performance space.

As an elitist algorithm, NSGA-II only maintains the information from the non-dominated guesses through each generation, or, in the case of the first iteration, from the first two generations. However, there are several ways to evaluate or approximate non-dominance. A straightforward way is to sum the four objective functions to determine a sort of cumulative distance from the origin; this approach is shown in Eq. 2.9. This is commonly done to some degree in hydrological studies that employ more than one objective function (Barco et al., 2008; Madsen, 2000). Deb et al. rebuts that this approach is only valid for problems in which the Pareto front is convex in shape, such as it is shown in Figure 1.3. Deb et al. provides a set of mathematical expressions to determine non-dominance (Deb et al., 2002), but these were neglected in favor of the first cut approximation of the Pareto front for the initial employment of the NSGA-II algorithm in SWMMCALPY. The full set of non-dominance expressions, as well as an outline for how that search would be conducted by a python script, is provided in Appendix D.

$$F = \sum_{i=1}^{M} w_i f_i \tag{2.9}$$

In Eq. 2.9, f_i represents each objective function (i.e. Eq. 2.1, 2.2, 2.7, and 2.8) and w_i is the corresponding user-defined weight on that objective function. F is the aggregate function by which each guess is evaluated.

For each iteration of NSGA-II, the union set $P \bigcup Q$ is sorted from lowest to

highest values of Eq. 2.9. The worse performing 50% of the union set (i.e. ranks 101-200) is culled and their temporary input files deleted. The surviving guesses form the new-initial set, P. For successive iterations of NSGA-II, the surviving guesses are mutated to create the younger generation, Q. SWMM simulations of guesses in set Q are evaluated by Eq. 2.9, the union set $P \bigcup Q$ resorted, and the worse performing half of guesses culled. This workflow of mutate, simulate, combine, sort, cull is repeated for many generations. NSGA-II is considered an elitist algorithm because the fittest guesses fill in the top ranks of the set P for each generation. While the information within these guesses are mutated to produce subsequent generations, the fit guesses themselves are preserved so long as they remain in the top half of $P \bigcup Q$. A helpful metaphor might be that the guesses within P are the reigning champs at the end of each iteration; they can only be removed in the next iteration if one of their children, in set Q, supplants them.

The final requirement for the successful execution of the NSGA-II algorithm is that a convergence criterion be established. Theoretically, each successive generation of guesses should more and more accurately predict the true Pareto front for the specific problem, but there is no stop condition built into the algorithm itself. Different researchers have investigated different criteria for halting the progression of the NSGA-II algorithm. Shinma and Reis used 100 iterations and concluded that was sufficient (Shinma & Reis, 2011). Deb et al. suggests a maximum of 25,000 function evaluations (Deb et al., 2002). With a population size of 100 sets, this corresponds to 250 generations. Others have used a moving window of the previous 500 iterations to determine whether the objective functions had substantially improved within that window (Krebs et al., 2013). Yet another study proposed the algorithm stopping criteria be respective of each objective function (Herrera

et al., 2006). The used of an aggregate objective function for NSGA-II sorting in the initial development of SWMMCALPY suggested that a simple generation count limit be implemented as the stop criteria; a value of 100 iterations was chosen. Other convergence criteria ideas, such like that presented in Shan and Wang (Shan & Wang, 2005), are explored in Appendix D. These convergence criteria are largely predicated on the NSGA-II sorting algorithm used, whether it is simplified or the full non-dominance sort proposed by Deb et al. (Deb et al., 2002).

The NSGA-II algorithm within SWMMCALPY is able to efficiently estimate the Pareto front for a given calibration problem. The list of contributing subcatchments, feasible parameter space, observational data timeseries, and equations for the objective functions are the arguments required to execute this portion of SWMMCALPY. The NSGA-II algorithm is not capable, however, of comparing solutions along the Pareto front.

2.5 Isolate Optimal Solution with User-Defined Weights

Weights on the four objective functions, as defined by the engineer at the outset of the SWMMCALPY routine, can be used to identify a single solution to be returned to the user. The available set of solutions is contained within the final set of guesses, P, that exists when NSGA-II has reached its stop criteria. This set, P, is an approximation of the true Pareto front that exists for the given calibration problem and therefore should be exclusively comprised of non-dominated solutions that are difficult to compare.

$$\sum_{i=1}^{n} w_i = 1 \tag{2.10}$$

Constraining the weights by Eq. 2.10 is possible because of the way the objective functions were formulated. It was posited that each objective function is bounded between [0, 1]. Bounding the weights similarly enables direct comparison between weighted objective function values, as they are likely to be of the same order. A weighted summation of unity was chosen arbitrarily; some consideration was given to each weight's value being on the order of one, but Eq. 2.10 has precedent in Lagrange multipliers commonly used for optimization.

For the aggregate objective function being used as the first cut of the NSGA-II sorting algorithm in SWMMCALPY, the objective function weights feature explicitly. At the conclusion of NSGA-IIs iterative process, the parameter set occupying the first-ranked position within set P is the optimal solution.

SWMMCALPY's composite python scripts can be found in Appendix C. Examples of the default and temporary files supporting the python scripts can be found in Appendix B, with descriptions. Appendix D contains a discussion about an alternative method by which the optimal solution can be chosen from the final pareto-approximating set. This method is designed to be compatible with the full non-dominance rank-sorting algorithm posed by Deb et al. (Deb et al., 2002) and the stagnating elitist set stop criteria presented by Shan and Wang (Shan & Wang, 2005).

Chapter 3: Test Case

While Example1.inp was used to aid in the building of SWMMCALPY's scripts, its limitations were numerous. SWMMCALPY is intended to be used by engineers, not solely as a research exercise. In that vein, a non-physically real, relatively small system like Example1 was not a robust enough test for SWMMCALPY's algorithm. For a better test case, a professionally developed SWMM model was used. To echo §1.1, the hypothesis is that SWMMCALPY's calibrated solution will perform comparably to the calibrated solution determined by an established method.

3.1 Brentwood: Model Description

The Brentwood neighborhood in northern Austin, Texas has experienced chronic flooding in the past few years when subjected to even mild storms (Geosyntec, 2017). The City of Austin recruited the consulting firm Geosyntec, Inc. to investigate the root cause of the flooding and propose a cost-effective solution. Geosyntec, Inc. built and calibrated an existing SWMM model using the PCSWMM software. A map of the Brentwood site is included in Figure 3.1, whereas an EPA SWMM rendition of the input file and model complexity information are included in Figure 3.2 and Table 3.1, respectively.

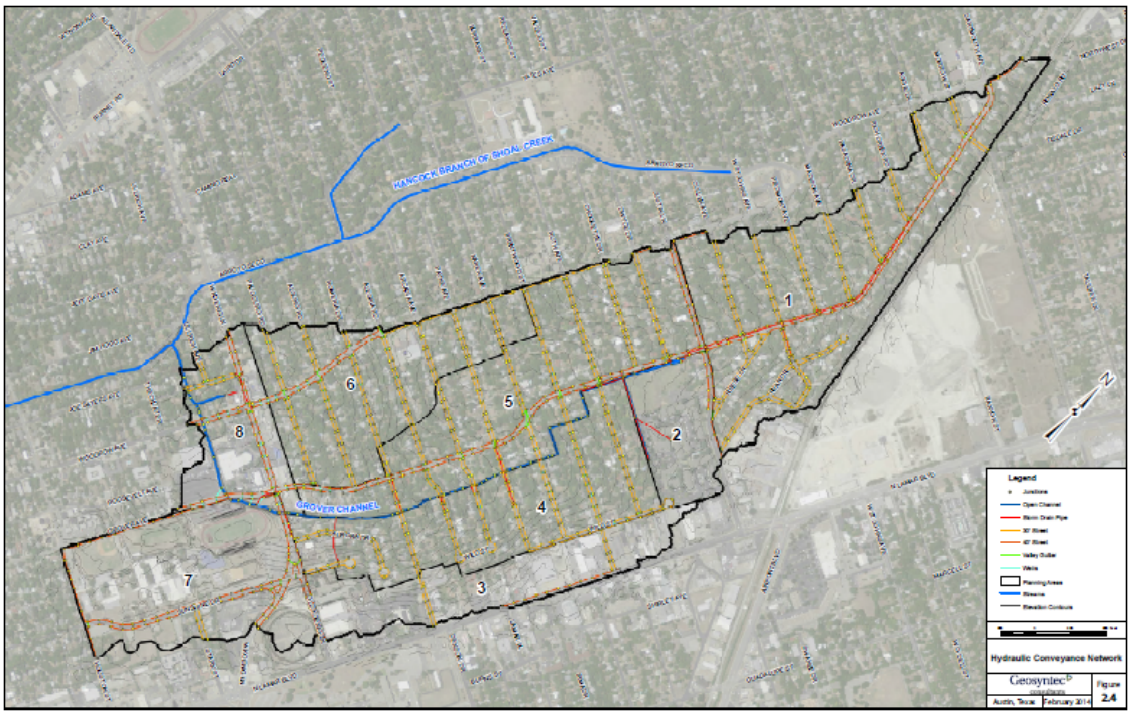


Figure 3.1: Map of Brentwood Case Study Extents. From Geosyntec (2017).

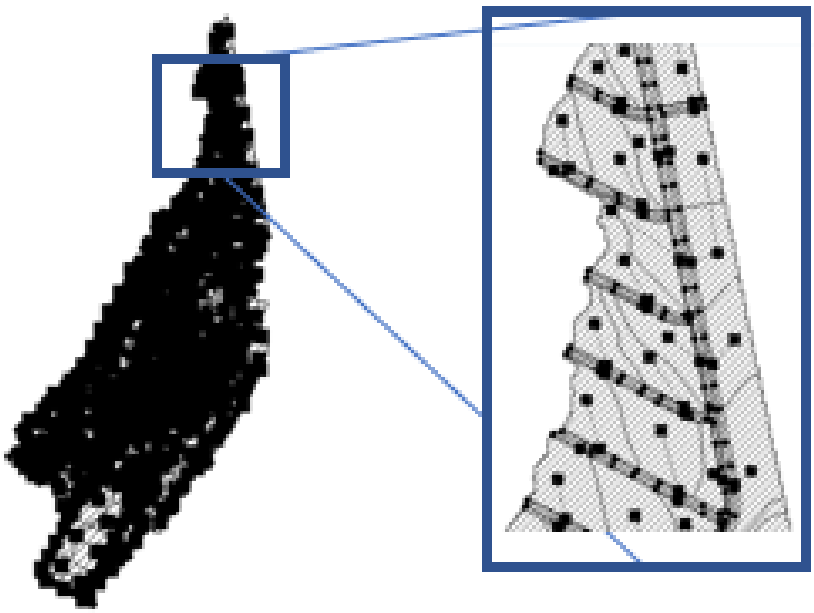


Figure 3.2: EPA SWMM 5.1 interpretation of existing Brentwood model, partially expanded so show detail of SWMM model.

Table 3.1: Model Complexity Data obtained from Geosyntec (2017).

Summary Statistics of Model Elements	Total
Total Area (ac)	368.3
Initial Impervious Cover (%)	45.7
# of Subcatchments	619
# of Junctions	1,221
# of Inlets	57
# of Conduits	1,355
Conduit Length (ft) – Pipe/Culvert	9,090
Conduit Length (ft) – Curb/Gutter	106,231
Conduit Length (ft) – Channel	5,934

Figures 3.1, 3.2, and Table 3.1 show that the Brentwood neighborhood model is a substantially more complicated system than Example1 and therefore poses a more significant and compelling litmus test of SWMMCALPYs algorithm.

The Brentwood system was built in PCSWMM and calibrated with PCSWMMs SRTC tool, further adding value to the test case (Geosyntec, 2017). Should SWMMCALPY manage to yield a calibrated solution for the Brentwood system that is comparable in performance to that which was obtained by professional engineers using a popular and effective approach, confidence will grow in SWMMCALPY's capacity to satisfy its objectives.

The Brentwood test case underwent several preprocessing steps to prepare it to undergo calibration via SWMMCALPY.

1. Prepare an "uncalibrated" copy of the model.

2. Select a single storm event.

3. Determine the objective functions on calibrated SWMM model.

3.2 Prepare an "uncalibrated" copy of the model

The Brentwood SWMM model input files provided by Dr. Brandon Klenzendorf of Geosyntec, Inc. were the calibrated and validated files used for the composition of their technical report (Geosyntec, 2017). To make running SWMMCALPY on the Brentwood case a useful exercise, a copy of the model first had to be "uncalibrated". Table 3.2 in the technical report provides a summary of changes made to the calibrated parameters by the SRTC tool. The table is presented as Table 3.2 here for convenience.

Table 3.2: Summary of Subcatchment Parameters Analyzed for Calibration obtained from Geosyntec (2017).

Parameter	Initial Value	Range	Calibrated Value
Subcatchment Width (ft)	Varies	Varies	-62% for non-ROW[1] -44% for ROW
Impervious Cover (%)	Varies	Varies	-11% for non-ROW[2] -5% for ROW[2]
N for Impervious Cover	0.016	0.008 to 0.032	0.029
N for Pervious Cover	0.2	0.1 to 0.4	0.35
Depression Storage for Impervious Cover (in.)	0.05	0.025 to 0.1	0.09
Depression Storage for Pervious Cover (in.)	0.07	0.035 to 0.14	0.123
Percent Impervious Cover with Zero Depression (%)	21.7	12.7 to 36.8	23.9
Percent Routed (%)	40 for non-ROW 0 for ROW	20 to 80	28.6 for non-ROW 0 for ROW
Suction Head (in.)	8	4 to 16	10.8
Saturated Hydraulic Conductivity (in/hr)	0.4	0.133 to 1.20	0.27
Initial Moisture Deficit	0.2	0.1 to 0.4	0.37

From Table 3.2, it can be seen that Geosyntec's scope for subcatchment parameter calibration was slightly different than that of SWMMCALPY's. "Percent Routed" and Green-Ampt infiltration parameters, "Suction Head", "Saturated Hydraulic Conductivity", and "Initial Moisture Deficit", were calibrated in the professional calibration of Brentwood, but are neglected by SWMMCALPY. Conversely, SWMMCALPY calibrates the subcatchment average slope, but, as the slope was calculated from GIS tools for Brentwood SWMM model development, Geosyntec held it static. The intersection of the two method's calibratable parameters were corrected to produce the "uncalibrated" input file to undergo re-calibration. Where available from Table 3.2, the initial values prior to SRTC

calibration were reinstated (this happened to correspond to all of the "[SUBAREAS]" parameters). For "Subcatchment Width" and "Impervious Cover(%)", the negative average of the calibration change in column four (ignoring the difference between ROW and non-ROW subcatchments) was reapplied to the calibrated value to approximate the initial value.

For example, for a given, hypothetical, calibrated subcatchment width value,

$$
\begin{aligned}
&Calibrated\ Width(ft) = 425 \\
&Average\ Change = \frac{-0.62 + -0.44}{2} \\
&Initial\ Width(ft) = (1 - Average\ Change) * Calibrated\ Width \\
&Initial\ Width(ft) \approx 650
\end{aligned}
\tag{3.1}
$$

The simple arithmetic in Eq.3.1 was applied to the "Subcatchment Width" and "Impervious Cover(%)" parameters for each subcatchment, of which there were many.

This coarse process of undoing Geosyntec's work calibrating the Brentwood file yielded a SWMM input file that could be fed as an input into SWMMCALPY for recalibration.

3.3 Select a single storm event

As they are currently encoded in the initial development of SWMMCALPY, the objective functions used by the NSGA-II algorithm can only be applied to a single storm event. Namely, this restriction falls on the NPE. If Eq. 2.1 is recalled, only the maximum value from the timeseries is passed, so multiple storm event peaks would get washed out by the global maximum. There are potential fixes for this (such as establishing a time gap criteria between local maxima), but they have not yet been implemented. In lieu of a way to evaluate model performance in a continuous simulation, the test case of SWMMCALPY on Brentwood will began with a single storm event.

The storm event to undergo analysis was chosen by selecting the highest daily rainfall amount within the 2012 - 2014 period for which data was available. October 13, 2013 had the most rainfall for this 2 year period and, thus, formed the basis for the single event test case analysis.

A challenge for single storm modeling that does not exist for continuous simulation is that of spin-up time. Inaccurate initial conditions can affect model performance, so the model must be simulated far enough back that the initial conditions are sufficiently approximated. Using a continuous simulation of the calibrated SWMM model for Brentwood as the ground truth, a spin-up analysis was conducted for the October 13 storm.

Using forcing data clipped to a variable time increment before midnight on October 13, the calibrated SWMM model for Brentwood was simulated, up until 14:00 on October 13. Evaluations of the Nash-Sutcliffe Efficiency (NSE) index were used to determine the sensitivity of each simulation to changes in the time

increment. An equation for evaluating sensitivity is framed within Eq.3.2.

$$Sensitivity = \frac{\partial R}{\partial P} \tag{3.2}$$

In Eq.3.2, R represents the "model" used, or, in this case, the NSE evaluated between the continuous simulation and single event simulation. P represents the parameter being changed, in this case, the time increment prior to midnight on October 13, 2013 that is included in the single event simulation.

Multiple time increments of forcing data were simulated for the single event. The sensitivity parameter described in Eq. 3.2 was computed and plotted against P, the time increment. The regression is presented in Figure 3.3.

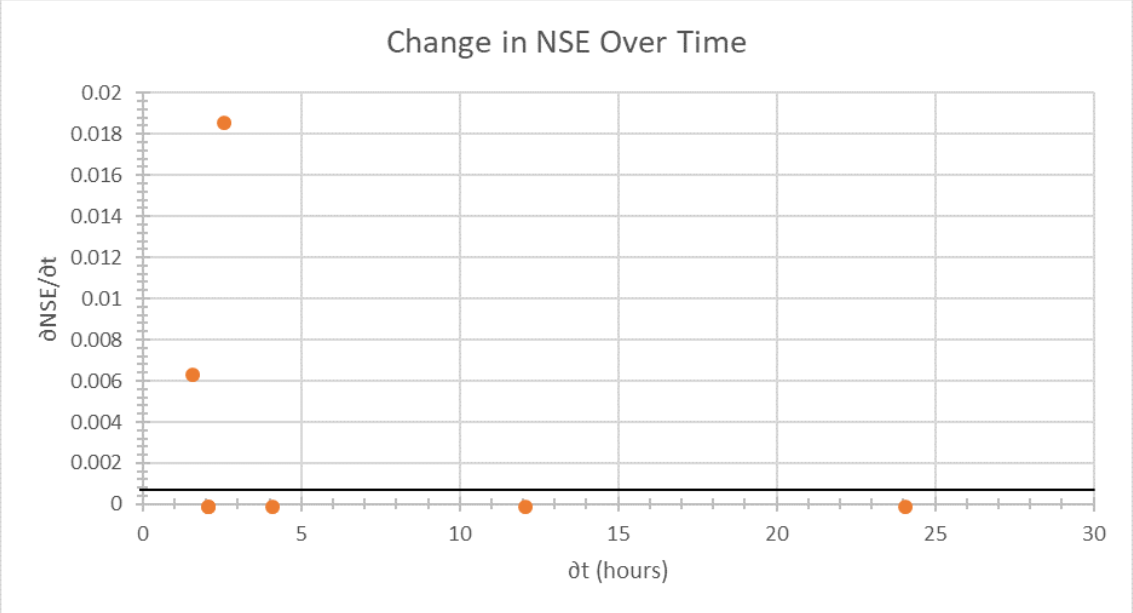


Figure 3.3: Sensitivity Parameter vs Time Increment for simulations with rainfall data clipped to several hours prior to 12AM on October 13, 2013. Copied from Appendix E.

From Figure 3.3 it can be seen that the sensitivity of the NSE to increased

time increment stagnates after roughly 10 hours. This indicates that only 10 hours of forcing data are required to produce as good as possible a single event simulation.

3.4 Determine the objective functions on calibrated SWMM model

With the time extent for a single event analysis established, the final pre-processing step for the Brentwood case is to determine how the calibrated SWMM model performs on the basis of SWMMCALPYs objective functions.

Observational data at the outfall of the system was clipped to the single event. SWMMCALPYs entire workflow was not necessary for this one set of calculations, so the observed data and SWMM input file were passed to the "ObjectiveFunctions.py" script (commented code provided in §C.4). This script evaluates Eq. 2.1 - 2.7. Table 3.3 contains the values of these objective functions for the October 13, 2013 storm in Brentwood.

Table 3.3: Objective Function Values of Brentwood model for Oct. 13, 2013

Objective Functions	
NPE	0.0939
NVE	0.2197
NSE_m	0.1495
NED	0.3182

October 13, 2013 represented the maximum rain event in the Brentwood model time frame, so the calibrated simulation may be further off for this event than other, milder storms. Even so, the calibrated Brentwood model performed

adequately for NPE and NVE and well for the NSE_m. These values were verified through a more transparent calculation with Microsoft Excel.

With the Brentwood model evaluated for its performance according to SWMMCALPYs goodness of fit criteria, the test case is prepared for comparison against SWMMCALPYs solution. If SWMMCALPY is able to manipulate the forcibly uncalibrated Brentwood file into a calibrated solution that challenges the benchmark set by the SRTC-calibrated model, the hypothesis of SWMMCALPY viability will be supported.

Chapter 4: Results

As is detailed in §2.5, an aggregate function, Eq. 2.9, was used as the ranking criteria for all of the guesses in a given generation. The top performing half of the guesses were maintained for future generations and the #1-ranked guess after the last iteration is returned to the engineer as the calibrated solution. Figure 4.1 is a plot tracking the value of the aggregate function of the #1-ranked guess in each generation. The #1-ranked guess was tracked for 5 runs of the NSGA-II algorithm.

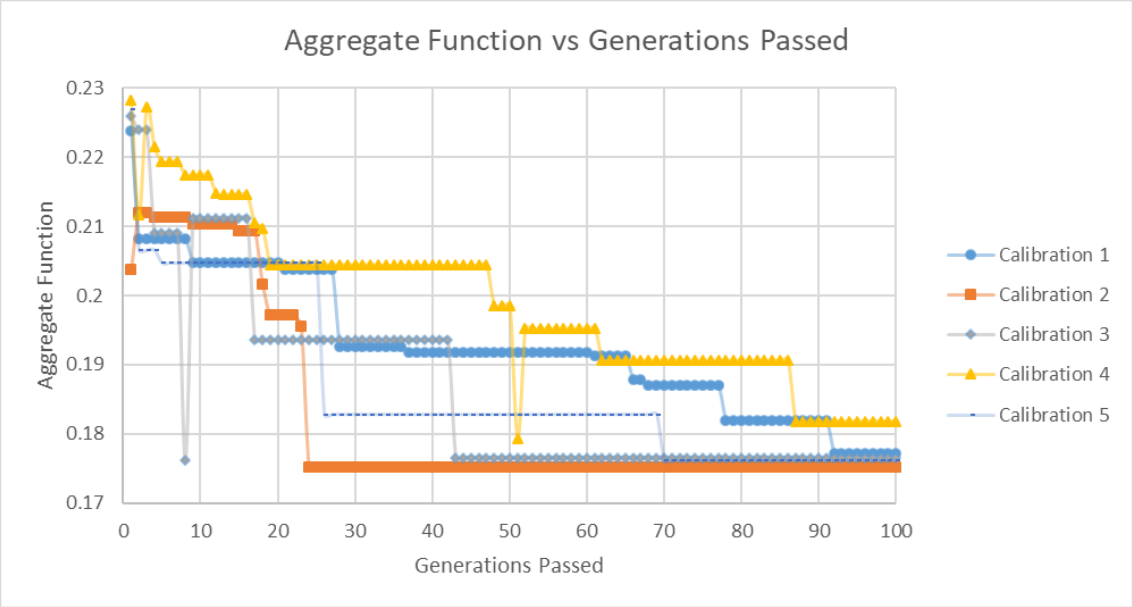


Figure 4.1: Time series showing the progression of the top-ranked guess in each of 100 generations.

Figure 4.1 is a generation series of the top-ranked guess for 5 runs of SWMM-

CALPY on the Example1.inp file. The weights used are given as:

$$w_i = \frac{1}{4} \tag{4.1}$$

which satisfies the constraining condition given by Eq. 2.10. All objective functions being weighed equally has the effect of distilling the aggregate function in Eq. 2.9 down to simply the average of the individual objective functions. The total inflow at the outfall of the Example1 system (node "18") was reported and compared against the synthetic "observed" dataset detailed in §B.2.

Figure 4.1 follows the predicted behavior wherein the aggregate function is steady at a given value, then jumps to a lower value once a more favorable parameter set is found. Interestingly, in several of the SWMMCALPY runs there are anomalies where one generation has a much lower aggregate function value, but that low value does not survive to the following generation. Upon further investigation, the anomalies arise when a new guess is promoted to the top rank. This is intuitive, as it was promoted for the very reason of having a lower aggregate function value. But, in the following generation, without the parameter set comprising the guess changing in any way, the aggregate function has a different, higher, value. This suggests that the bug may have its roots in an inconsistency in the objective function formulations themselves. Addressing this coding bug will be a priority, detailed in §5.2.

In Figure 4.2, it can be seen how each objective function behaved through the generations. Figure 4.2 was included to highlight how the modified NSGA-II algorithm ranked the aggregate function, which was an average of each objective function. So it was possible for the best guess in each generation to switch dra-

matically between performing well on, say, the NED to performing well on the NVE. Table 4.1 then details the values of the objective functions for the solution returned to the user for each of the calibration runs.

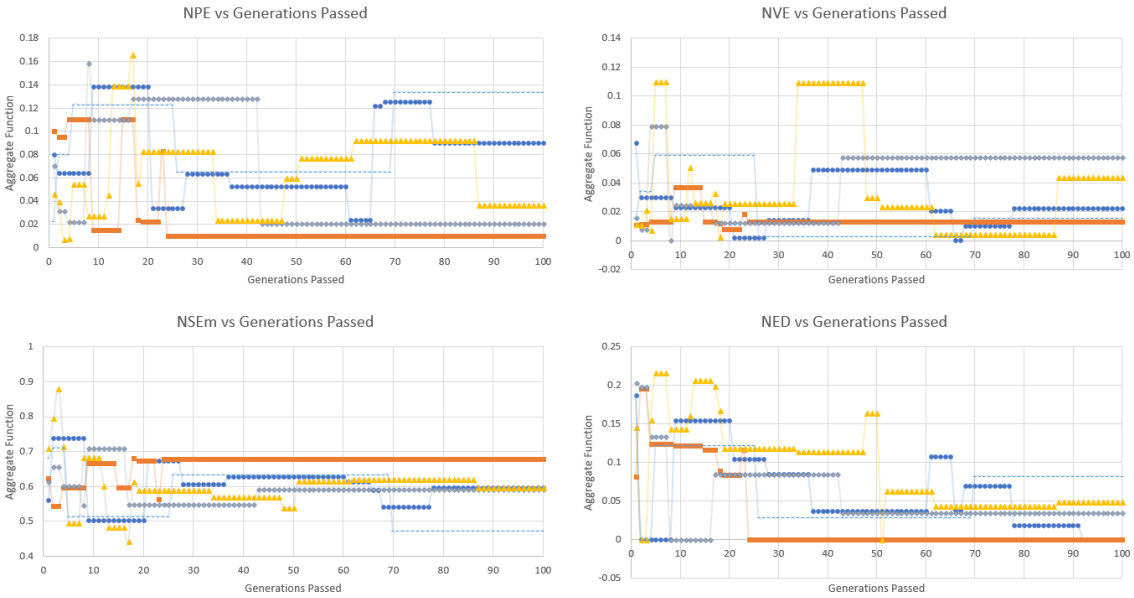


Figure 4.2: Time series showing the progression of each objective function for the top-ranked guess through 100 generations.

Table 4.1: Multiple objective function values of solutions for several runs of SWMMCALPY.

	NPE	NVE	NSE_m	NED	Aggregate
Run 1	0.0900	0.0223	0.5967	0.0000	0.17725
Run 2	0.0099	0.0132	0.6778	0.0000	0.17523
Run 3	0.0209	0.0578	0.5925	0.0351	0.17658
Run 4	0.0370	0.0439	0.5971	0.0494	0.18185
Run 5	0.1339	0.0158	0.4721	0.0827	0.17613

Something to notice is that all of the calibration runs tended to converge

to a value of about 0.17-0.18. Additionally, the values for the NPE, NVE, and NED were consistently an order of magnitude less than the NSE_m values. The NSE_m was added to the suite of objective functions for its ability to capture the general shape of the hydrograph, with emphasis on the timing of peaks. Figure 4.3 shows how the modified NSGA-II algorithm fails to capture the timing of the synthetic observed data for calibrated solutions of the Example1.inp system.

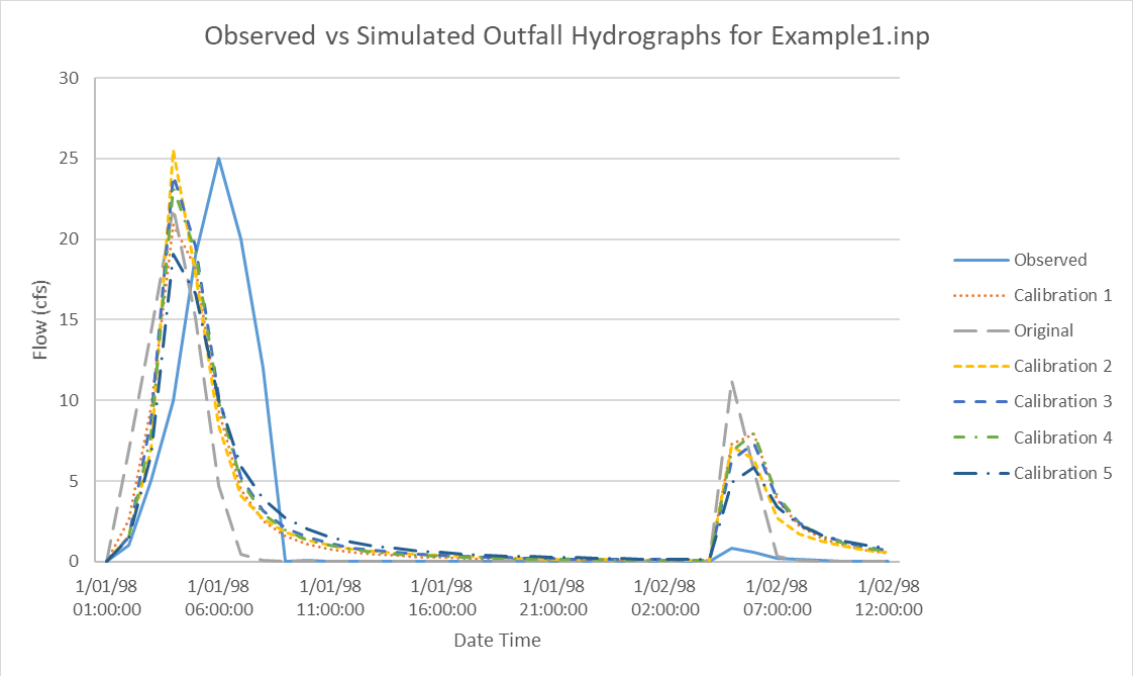


Figure 4.3: Hydrograph comparisons of synthetic observed data, the original Example1 system, and several calibrated solutions from SWMMCALPY.

The original Example1 system and each calibrated solution have the same peak timing, which is two hours early on the peak for the synthetic observed data. A possibility to consider is whether the synthetic observed data (which was produced before any calibration was done) is even a realizable behavior for the given SWMM model. It is possible that no parameter value configuration of the

small, simple Example1 SWMM model can delay the time of concentration enough to reproduce the hydrograph behavior contained in the synthetic observed data.

Potentially a more consistent and revealing test for SWMMCALPY with the Example1 system,a model-model comparison will be explored. Two independent sets of parameters are constructed as the starting position and "ground truth". The "observed" data simply becomes the hydrograph from a simulation of the ground truth data set, thus guaranteeing that the observed data is at least realizable from the model.

Chapter 5: Conclusions and Future Work

5.1 Conclusions

The problem statement motivating SWMMCALPY presented at the outset of this thesis was to develop a routine that automates the calibration of the Storm Water Management Model by making use of a genetic algorithm. Three sub-objectives: defining a variable calibration scope, determining the Pareto front for multiple objective functions, and isolating the optimal solution for user-defined weights, were executed to create a natural outline for SWMMCALPYs workflow, as shown by Figure 1.1. An simple SWMM system, Example1.inp, was used as a experimental trial proof of concept for SWMMCALPY.

Defining a variable calibration scope allowed the engineers to relax the constraint of perfectly correlated parameter values across the system. The NetworkX package and recursive search allowed for all contributing nodes to a point to be identified. This method is flexible enough to capture simple tree systems, as well as cross-cutting systems, interconnections, and extraneous outfalls.

Multiple objective functions that address independent aspects of a hydrograph were attempted to be minimized. To balance guesses that compromised on one objective function to minimize another, an aggregate function that averages the objective function values was employed. A modified NSGA-II algorithm evaluated the fitness of each guess and mutated the fittest guesses to approach an approximation of the Pareto front for a given calibration system. After 100 iterations of the NSGA-II algorithm, the top-ranked guess was returned as the

solution to the engineer.

In Chapter 4, a comparison of several runs showed that the algorithm was capable of yielding a variety of solutions with noticeably different behaviors, but similar aggregate function values. This highlights the many-to-one issue with multiple objective functions and invites a comparison with the full-blown NSGA-II search algorithm detailed in Appendix D.

For simple systems, like the Example1.inp model, SWMMCALPY shows promise as a viable and flexible calibration routine. However, there is an issue with routine run time with larger systems. The Brentwood neighborhood test case analysis was not included in this thesis because of the burgeoning computational requirements as system complexity increases. To illustrate, 100 generations of the modified NSGA-II algorithm, with a generation size of 100 guesses, requires 10,000 SWMM model simulations. Run serially, this is easily the most time intensive aspect of the SWMMCALPY workflow. Some manner of parallelization for will be critical for expanding the scope of calibratable problems with SWMMCALPY.

5.2 Future Work

Despite success in yielding a calibrated solution for a simple SWMM system, SWMMCALPY remains in its infancy. More analysis, combined with myriad fixes and upgrades are necessary to make the routine usable by the general SWMM community. This upcoming summer, the author will continue to work on SWMM-CALPY to address the fixes discussed in previous sections and here.

In Figure 4.1, a strange anomaly in the progression of the top-ranked guess in each generation arose. The fact that the bug seemed to be when a guess had an inconsistently low value for the aggregate function and then restabilize at a more

consistent value, points to the issue being with the evaluation of the objective functions themselves. So, the search for the bug will begin with the "Objective-Functions.py" file contained in §C.4.

SWMM simulations themselves consume the preponderance of the time taken by the SWMMCALPY routine. Currently, each SWMM simulation for a given generation is computed in series, which essentially renders SWMMCALPY unusable for realistically complicated SWMM models. However, there is no logical reason for each SWMM simulation in a generation to be computed in series, which means that there is potential for a two order of magnitude reduction in the computational time to conduct a SWMMCALPY run. Parallelization of SWMMCALPY will be explored and an effort will be made to bring a problem the size of the Brentwood neighborhood test case described in Chapter 3 into the plausible scope of the routine.

Another time saving feature that will be investigated in the future development of SWMMCALPY is the inclusion of a sensitivity analysis preceding the actual calibration phase. As alluded to in §1.3.2, sensitivity analysis has the affect of reducing the dimensionality of the calibration problem. This won't, on it's own, contribute to speeding up the algorithm, but it could potentially increase the rate of convergence to the Pareto front as insensitive parameters are neglected. An increased rate of convergence could allow for fewer generations being evaluated to achieve the desired accuracy. The post-processing corollary to sensitivity analysis is uncertainty analysis. Several ready-made python packages exist to conduct analyses like the generalized likelihood uncertainty estimation(GLUE; described further in §A.9). Uncertainty analysis would give the engineer an understanding of the variance of individual parameters among Pareto front solutions.

In addition to parallelizing some aspects of the genetic algorithm component of SWMMCALPY, several components of the modified NSGA-II sorting algorithms stand to be upgraded. Appendix D contains details of how a more sophisticated Pareto front-nearness search algorithm can replace the aggregate function used in Chapter 2. A comparison between the two search methods could yield fruitful information about just how sensitive the solution is to the search method for multi-objective optimization problems.

Potential for improvement in generalizing the variable calibration scope problem also exists in this version of SWMMCALPY. §2.3 introduces a challenge posed by multiple locations of observational data within a system, especially when the contributing subsystems to that location are overlapping. Solving that problem will likely involve ramping up the weight for the NED objective function for those subcatchments which are involved in both clipped subsystems. The logic is as follows. When one calibratable subsystem exists, the confidence in all subcatchment parameters is the same. However, when additional observational data causes overlapping subsystems, it is not reasonable to overwrite the calibrated parameters from the first calibration problem, nor is it reasonable to assume they will not be affected by additional information. The most reasonable solution is to increase the "confidence" in the solution of the subcatchments that have already been calibrated through an increase in the weighting of the NED objective function for those subcatchments. This proposed solution to an overlapping calibration problem needs to be investigated further.

These problems and opportunities for improvement frame the direction of future development of SWMMCALPY. The overarching goal was to produce an automated calibration algorithm that was useful to SWMM users; SWMMCALPY

has potential to live up to that expectation. However, improvements to the algorithm and the code itself are a necessary step towards that aim.

Appendices

A Automated Calibration of the EPA's Storm Water Management Model: A Review of Published Works

A.1 Introduction

The Storm Water Management Model (SWMM) has been under development by the U.S. Environmental Protection Agency (EPA) since 1969. It has since undergone nearly 50 years of evolution, taking the form of 5 major versions (Huber & Roesner, 2012). SWMM is a physically based model, similar to the Army Corps of Engineers-supported Hydrologic Engineering Center - Hydrologic Modeling System (HEC-HMS). Physically based models are best described by contrasting them with their foil, "black box" models. Black box models consider the catchment under scrutiny to be an opaque control volume, with precipitation as the input, and runoff as the output. Examples of black box models are unit hydrographs that relate a hyetograph directly to the marginal runoff, and a simple convolution will yield the storm-specific hydrograph (Wang et al., 2012). SWMMs approach is to have a transparent control volume and attempt to make predictions about the commonly sought outcome (the behavior of the runoff) by describing the physical process that occur between, and ultimately affect, the rainfall-runoff relationship. The processes occurring within the control volume of the watershed can include evapotranspiration, infiltration, and storage, each of which has a physical consequence on the amount of runoff that will occur from a given precipitation event. Subsequently, engineering decisions that affect these physical processes are highly investigated for their influences on the all-important rainfall-runoff relationship. This strain of inquiry forms the basis for the preponderance of scientific studies that utilize SWMMs modeling capacity (Cipolla et al., 2016; Feng & Burian, 2016;

Guan et al., 2015b; Rosa et al., 2015; Tobio et al., 2015).

SWMMs strength, the capacity to use physical processes to describe the behavior of water in a catchment, is also a point of weakness, as it requires an increased number of parameters that need to be assigned value accurately in order to effectively match the observed runoff behavior. Compounding this, many of these parameters, such as Manning's roughness or infiltration coefficients, are virtually unknowable at any scale larger than a laboratory setting. Rigorous scientific research has yielded ranges for these variables, many of which can be found in the SWMM user's guide written by James, Rossman, & James (Rossman et al., 2010) or other such handbooks (Maidment, 1993). For the vast majority of scientific research, these ranges do not offer the precision necessary to make useful predictions, so some level of calibration is needed (Guan et al., 2015b; Krebs et al., 2013; C. Li et al., 2016; J. Li et al., 2016).

The purpose of this report is to investigate which calibration methods are commonly used in published works that employed SWMM for their analysis. A special focus was placed on *automated* calibration methods, and the differences between them, with the ultimate goal of recommending a calibration procedure that can be supported by the SWMM program for future standardization. According to Rossman (Rossman et al., 2010), there is currently no automated calibration procedure recommended by SWMM version 5.

A.2 "Calibration" broken down

Rossman (Rossman et al., 2010) described calibration as more than just one step. The common term "calibration" is broken down into sensitivity analysis, actual calibration, and uncertainty, or error, analysis. However, the investigation that

yielded this report could not find a single article that indicated all three steps were conducted; some do the first two steps of sensitivity analysis and calibration in an automated fashion (Krebs et al., 2016), while still others do these steps, but some portion of the analysis is done manually (Knighton et al., 2014; C. Li et al., 2016).

☐ Sensitivity Analysis

☐ Calibration

☐ Uncertainty Analysis

This checklist creates a natural outline for this report. If articles that demonstrate the presence of more than one of these steps are considered to be more sophisticated than those that did one or fewer, and articles that show that the steps were done with the assistance of automated algorithms are considered to be more sophisticated than those that conducted the analysis manually, a research ranking system emerges. The remaining sections of this report step through this ranking system, commenting on the research done and the methods used in each sophistication tier. To reiterate, the primary objective of this literature review is to identify methods that could be potentially be adopted as the default methodology for SWMM calibration. Stepping through different levels of research sophistication and the methods used in each one will help cultivate a sense of which methods are used in which contexts, and whether any other usage patterns emerge. Papers using less sophisticated calibration methods were included in this discussion so that a holistic understanding of the body of work surrounding SWMM could be achieved.

A.3 No Calibration Steps

Due to the difficulty in predicting, to an acceptable degree of precision, the values and rates that govern the physical processes occurring within a unique catchment, very few published articles attempt to model a study catchment without any calibration. However, Shen and Zhang Shen and Zhang (2014) did just that in their analysis of catchments based on GIS and remote sensing data. Their objective was to determine whether or not the catchment could be reasonably described by computed physical parameters like those used by Horton's equation. For this reason, their lack of calibration was justified, albeit unhelpful in terms of illuminating a useful automated calibration method.

In 2016, Feng and Burian (Feng & Burian, 2016) tried a different approach. They simply trusted the rigor of their literature review to give them parameter values that would reasonably match the rainfall-runoff relationship shown in their observations. Their ultimate conclusion was a "fair" agreement between their model and the observed relationship.

A.4 Manual Calibration

The second lowest sophistication tier contains research papers and bodies of work that conducted the minimum level of calibration on known rainfall-runoff relationships to make useful predictions about what they might look like after the catchment has undergone some change. Invariably, these studies first conducted a literature review to get ranges for the parameters they wished to calibrate. Often among these cited works was Rossman (Rossman et al., 2010). These articles also tended to cite one another. Studies that employed calibration of any kind

greatly improved their modeling accuracy over studies that neglected it (Tobio et al., 2015). Manual calibration involves adjusting model parameters one at a time until a certain level of satisfaction that the model matches the observed relationship is attained. A validation step to ensure the calibration retains accuracy across multiple events or time scales almost always follows. Achieving this level of satisfaction often takes the form of maximizing the Nash-Sutcliffe Efficiency index (NSE) (Guan et al., 2015a, 2015b; Rosa et al., 2015; Tobio et al., 2015), although occasionally it is done by some other method (Cipolla et al., 2016).

$$NSE = 1 - \frac{\sum_{t=1}^{T}(Q_0^t - Q_m^t)^2}{\sum_{t=1}^{T}(Q_0^t - \bar{Q}_0)^2} \tag{A.1}$$

The NSE is a measure of the difference between the calibrated model's prediction and the observed runoff behavior. Maximizing the NSE means reducing this difference as much as possible (Nash & Sutcliffe, 1970). Typically, values of $NSE > 0.85$ are considered good (Versini, Ramier, Berthier, & de Gouvello, 2015).

Tobio et al. (Tobio et al., 2015) used the NSE to determine optimal design parameters for Low Impact Development (LID) in Korea. Experimental LID's were tested, and their hydrological parameters determined by matching their modeled runoff behavior with the observed hydrograph.

A.5 Manual Sensitivity Analysis & Calibration

In this next tier, it becomes useful to discuss the value in coordinating the calibration and sensitivity analysis steps. The way sensitivity analysis is intended to be carried out is as a mechanism to simplify, or inform, the calibration step.

By determining which parameters affect the predicted hydrograph the most, the calibration can place emphasis on those parameters and afford to neglect the others. Especially when the calibration step is being done manually, this can save the researchers valuable time (Guan et al., 2015a, 2015b; Rosa et al., 2015).

Although the manual calibration step is dominated by the NSE, researchers have used various methods when it comes to evaluating the sensitivity of their model to incremental parameter adjustments. Rosa et al. Rosa et al. (2015) used a method to evaluate relative sensitivity.

$$Sensitivity = \left(\frac{\partial R}{\partial P}\right)\left(\frac{P}{R}\right) \tag{A.2}$$

Where ∂R is the difference between the original and new model output in response to a ∂P change in the parameter value, and R and P are the original model output and parameter values.

This method is not common, however, and other methods have received more widespread attention for use in sensitivity analysis. For example, Guan et al. (Guan et al., 2015a, 2015b) used the NSE for both their sensitivity analysis as well as their calibration. A parameter can be considered sensitive if a marginal change in the parameter yields a large response in the model output relative to the responses of other parameter changes. In this case, the NSE wouldn't have to be maximized, but the objective is to find those parameters for which $\frac{\partial NSE}{\partial P}$ is great.

A.6 Automated Sensitivity Analysis with Manual or No Calibration

Operating under the notion described in §A.5, wherein the labor-intensive process of calibration can be streamlined with an effective sensitivity analysis step, many studies spend significant effort to that end (Knighton et al., 2014; C. Li et al., 2016; J. Li et al., 2016).

Li et al. (C. Li et al., 2016; J. Li et al., 2016) both used a modified Morris screening method for their sensitivity analysis, obtained from (Campolongo & Braddock, 1999). The modified Morris screening method is not unlike the Nash-Sutcliffe Efficiency Coefficient when used for this purpose. They both relate the change of the model output to a marginal change in a single parameter.

$$S = \left| \sum_{i=1}^{n} \frac{(Y_{i+1} - Y_i)/Y_i}{(P_{i+1} - P_i)/P_i} \Big/ n \right| \tag{A.3}$$

The difference is that the modified Morris screening method can be coded so that it proceeds automatically and returns the variables ranked in terms of their "S" value, while the NSE requires some level of oversight. Li et al. (C. Li et al., 2016; J. Li et al., 2016) then proceeded to use the NSE to calibrate their most sensitive parameters.

Knighton et al. (Knighton et al., 2014) introduced way of thinking about sensitivity analysis that is unlike any of the other studies discussed to this point; the distinction between deterministic and stochastic sensitivity analysis. Both the modified Morris screening method and the Nash-Sutcliffe Efficiency Coefficient yield *a number*, indicating that each parameter affects the model outcome in a

quantifiable, predictable, and consistent way. Knighton et al. (Knighton et al., 2014) argued that this "deterministic" approach to the sensitivity analysis is inappropriate and biased, and that a better way to conduct sensitivity analysis is through a probabilistic, or stochastic, approach. In his research, Knighton et al. (Knighton et al., 2014) preferred a Monte Carlo-style analysis of model sensitivity, which, essentially, says that each parameter has a Gaussian distribution of potential sensitivities. It is worth noting that stochastically described parameter sensitivities can still be related deterministically by comparing the means of their probability distributions. Knighton et al. (Knighton et al., 2014)'s ideas about stochastic sensitivity analysis were echoed in his views of the uncertainty analysis portion of the calibration process, and will be discussed later.

A.7 Automated Calibration with Manual or No Sensitivity Analysis

This section is the converse of §A.6, focusing on research studies that prioritize the calibration step over the sensitivity analysis step. This can be afforded because automatic calibration algorithms are employed, cutting down the labor-intensive process that the researchers mentioned in the former section were also looking to avoid. It is in this section that many of the processes that may be viable for official adoption by SWMM are introduced.

Barco et al. Barco et al. (2008) is one of the most well-cited works in this field with 97 citations. The introduction section of this research article is a miniature literature review in and of itself, shedding light on the articles that gave rise to many of the most common automated calibration methods in use today. From the genetic algorithm that evolved into the genetic multi-objective optimization algorithm (NSGA-II) detailed by (Deb et al., 2002), to the artificial neural network

(ANN) method coined by (Zaghloul & Abu Kiefa, 2001) but based on the various works authored in part by Liong between 1991-1995 (Ibrahim & Liong, 1992; S. Liong & Ibrahim, 1994; S. Y. Liong, Chan, & Lum, 1991; S.-Y. Liong, Chan, & ShreeRam, 1995), Barco et al. (Barco et al., 2008) offered a nearly comprehensive history of automated calibration method options. Ultimately, the analysis conducted by Barco et al. Barco et al. (2008) and her team was done using the complex method put forth by Box (Box, 1965), but many studies investigated alternative methods enumerated by Barco et al. (Barco et al., 2008).

A concept that has been hinted at, but not yet formally defined, is that of a calibration target function. When the calibration was being done manually the most common target function was maximizing the NSE. This can be reasonably related to matching the total volume of the runoff response, due to the cumulative difference between the modeled and observed behavior term in the equation. Many studies, especially those that focus on the effects of urbanization on flooding, were more interested in changes to the peak flow rate, rather than total volume, as a result of urbanization or, conversely, some urbanization mitigation strategy (Barco et al., 2008; Krebs et al., 2013, 2014). Barco(Barco et al., 2008) and her team understood this, and so they developed a customized target function for their research.

$$F = w_1\left(\frac{Q^* - Q}{Q^*}\right)^2 + w_2\left(\frac{P^* - P}{P^*}\right)^2 + w_3\sum_{i=1}^{n}\left(\frac{Q^* - Q}{Q^*}\right)_i^2 \tag{A.4}$$

Where Q is the total flow volume, P is the peak flow rate, and f is the instantaneous flow rate. "w_1", "w_2", and "w_3" are weighting factors that let Barco et al. (2008)

and her team bias their calibration towards the part of the flow behavior that was most important to them at the time.

Despite Barco's (Barco et al., 2008) comprehensiveness in the literature review portion of the 2008 study, and her creativity in the calibration portion, parameters were selected for calibration without any consultation of a sensitivity analysis. After the calibration process was finished, a manual sensitivity analysis was conducted, seemingly as a discussion point.

Although the complex method was denounced as "precise but slow" by Masseroni and Cislaghi (Masseroni & Cislaghi, 2016), it is a reasonably popular automatic calibration method. Granata et al. (Granata, Gargano, & de Marinis, 2016) used it as well in his comparison of SWMM versus Support Vector Regressions for modeling rainfall-runoff relationships in urban areas.

Other studies utilized alternative algorithms to achieve the same goal. The NSGA-II method described by (Deb et al., 2002) was used by Krebs et al. (Krebs et al., 2013, 2014). In both of Kreb's Krebs et al. (2013, 2014) articles, the multi-objective calibration was preceded by a manual sensitivity analysis with the NSE as the objective function.

Another option for the calibration step is to involve another program. "fmincon" is a function available in MATLAB that uses gradients to approach the optimization. As with the NSGA-II method, a target objective is required, and the NSE is often used (Masseroni & Cislaghi, 2016).

In §A.6, it was discussed that Knighton et al. Knighton et al. (2014) made use of a class of functions, Monte Carlo simulations, to approach sensitivity analysis a different way. Similarly, the Rosenbrock methods are commonly used for solving differential equations problems, such as the ones posed in calibration. Versini et

al. (Versini et al., 2015) used the Rosenbrock methods for calibration of SWMM for the sake of analyzing the impacts of green roofs in a French basin. Wang et al. (Wang et al., 2012) also calibrated his SWMM model by employing this class of functions.

As a side note, Wang et al. (Wang et al., 2012) contrasted classic calibration, like using Rosenbrock, fmincon, or NSGA-II, with a method like ANN. Wang et al. (Wang et al., 2012) considered this method to be inappropriate for a physically based model such as SWMM because of the reductive nature of the method. ANN is described as a "learning algorithm" because it finds its parameters by matching hydrographs over multiple events. That is, after a while, a program utilizing ANN would learn to identify that a rain event with certain characteristics would probably produce a particular runoff result. Wang et al. Wang et al. (2012) disagreed with the use of this method because it has the effect of reducing the transparent, physically based, flexible SWMM model into a black box. To corroborate Wang et al. (2012)'s disapproval, no recent papers were found that utilized the ANN method for SWMM calibration.

A.8 Automated Sensitivity Analysis and Calibration

This section discusses articles that demonstrated the presence of automated sensitivity analyses followed by automated calibration. This approach would yield the least labor-intensive calibration of SWMM, provided the methods can be executed efficiently. Krebs et al. (Krebs et al., 2016) utilized the generalized likelihood uncertainty estimation (GLUE) for automatic sensitivity analysis, which was absent in his 2013 and 2014 papers (Krebs et al., 2013, 2014). GLUE, introduced by (Beven & Binley, 1992), works differently than the NSE and modified Morris

screening methods discussed previously. The latter methods use a relative system, wherein the parameters that affect the model result the *most* are the ones considered to be sensitive. The appropriate number of sensitive parameters to include was still subjective. GLUE standardizes and automates this process by introducing statistics. GLUE begins similarly to the other methods, by adjusting one parameter at a time and observing the severity in the change in model result. The difference is that GLUE checks whether that change in model result is statistically significant, thereby maintaining the sensitivity ranking system of the other methods, but establishing an objective threshold for sensitivity.

Krebs et al. (Krebs et al., 2016) followed his GLUE-driven sensitivity analysis with the same NSGA-II calibration technique employed in 2013 and 2014 (Krebs et al., 2013, 2014). This time, however, the objective functions used were the sum of the squared error and the volume error. Krebs admitted that the two objective functions yield appreciably different parameter values once calibrated, an issue which will be discussed further in the next section.

A.9 Automated Sensitivity Analysis and Uncertainty Analysis

This last section focuses on the most sophisticated studies. It is worth noting, and rather surprising, that none of the research articles consulted for this report clearly demonstrated all three Rossman-defined steps for the calibration process. That being said, Knighton et al. (Knighton et al., 2016), was by far the most sophisticated paper referenced. Knighton et al. (Knighton et al., 2016) built on his 2014 paper by revisiting the topic of the best approach for sensitivity analysis. He also discussed a methodology for uncertainty analysis which the majority of researchers shy away from, or deem unnecessary.

For the sensitivity analysis portion, Knighton et al. Knighton et al. (2016) elected to use the multi-objective generalized sensitivity analysis (MOGSA), which was first proposed by his partner, Bastidas, in 1999 (Bastidas, Gupta, Sorooshian, Shuttleworth, & Yang, 1999). This sensitivity method is novel to this report because, unlike the methods discussed previously, it allows for multiple objective functions *in the sensitivity analysis step.* Knighton et al. (Knighton et al., 2016) relied on a Pareto-ranking system for experimental objectivity.

Tackling the complicated and subtle task of parameter uncertainty analysis, Knighton et al. (Knighton et al., 2016) and his team chose to use the GLUE method that has previously been described as useful for the sensitivity analysis. Two primary considerations when using GLUE are the initial, or prior, distributions and the likelihood function used. In this case, Knighton et al. (Knighton et al., 2016) chose to select a uniform distribution across the feasible parameter space. The likelihood function used was chosen, after some deliberation, as the one proposed by Stedinger et al. (Stedinger, Vogel, Lee, & Batchelder, 2008).

$$L(\theta|D) = exp\left[-\frac{n}{2}\frac{\sum_{t=1}^{n}(D_t - D_t)^2}{\sum_{t=1}^{n}(D_t - D_{tMLE})^2}\right] \tag{A.5}$$

**It is worth noting here that this equation was copied exactly from Knighton et al. (2016), but there seems to be a typo in the paper, as (D_t-D_t)^2 will always be zero.

This likelihood function was coupled with the posterior probability of θ.

$$P(\theta|d_2, d_3) = k \times L(\theta|d_2) \times L(\theta|d_3) \tag{A.6}$$

These functions represent a "formal Bayesian approach", which Knighton et al. Knighton et al. (2016, 2014) highly preferred. Included in his article is an argument by Mantovan and Todini (Mantovan & Todini, 2006) and Mantovan et al. (Mantovan, Todini, & Martina, 2007) suggesting that using an informal function with GLUE, such as NSE, "represents an incoherent and inconsistent methodology" and "has no statistical basis".

The conclusions of Knighton et al. (Knighton et al., 2016) are also deserving of comment. The objective of the article was to determine whether or not there was a universal set of sensitive model parameters, rendering the whole sensitivity analysis process obsolete. Alas, their conclusion was otherwise: there is no universal set of sensitive parameters. This has implications of instilling doubt in other studies, like Barco et al. (Barco et al., 2008), which assumed sensitive parameters based on a literature review.

The other steps of the Rossman-defined calibration process did not suffer the same contention that seems to surround the uncertainty analysis community. Several other researchers have also weighed in on the generalized likelihood uncertainty estimation and its place in high-quality scientific work. Studies like (Zhang et al., 2015) and (Sun et al., 2013) argued the importance of the uncertainty analysis step and posit that GLUE can be effective if formal likelihood functions that reflect Bayes' Theorem of Probability are used, rather than informal, deterministic functions like NSE. Muleta et al. (Muleta et al., 2013) disagrees, saying that other

methods, like the Markov-chain Monte Carlo method, are better suited for uncertainty analysis. Mantovan et al. (Mantovan et al., 2007) is a response to Beven, the inventor of the GLUE method, after Beven made a critique of (Mantovan & Todini, 2006).

A.10 Recommendations

The objective of this report at the outset was to investigate the way that SWMM is being calibrated automatically in current literature, and to try to make recommendations about which of these methods is best. It might be useful at this juncture to recap the automated methods that have found popular use.

A.10.1 For Sensitivity Analysis

- Modified Morris screening method (C. Li et al., 2016; J. Li et al., 2016)
- Monte Carlo simulation (Knighton et al., 2014)
- Generalized likelihood uncertainty estimation - with Nash-Sutcliffe Efficiency Coefficient as the objective function (Krebs et al., 2016)
- Multi-objective generalized sensitivity analysis (Knighton et al., 2016)

Only one of these methods, MOGSA, affords the researcher the flexibility of multiple objective functions to define sensitive parameters. Much of the time, this flexibility exceeds the requirements of the study, as the researcher only cares about either the peak flow or the total flow runoff volume, but in the rare case that both, or additional runoff qualities, require scrutiny, MOGSAis the only method from

this list that can effectively differentiate between parameters that are sensitive for two different calibration objectives.

A.10.2 For Calibration

- Rosenbrock (Versini et al., 2015; Wang et al., 2012)
- fmincon (Masseroni & Cislaghi, 2016)
- Complex multi-objective (Barco et al., 2008; Granata et al., 2016)
- Genetic multi-objective optimization algorithm (Krebs et al., 2013, 2014, 2016)

Fmincon can be eliminated as viable, as it unnecessarily cumbersome to use an additional program to conduct an automatic calibration. The other three methods have achieved reasonable popularity and success, but Masseroni and Cislaghi's (Masseroni & Cislaghi, 2016) critique of the complex method is valid, so it too is removed from consideration. To decide between the who remaining methods, the same logic used in the sensitivity analysis discussion is applied here. The flexibility to bias a calibration to ensure accuracy on the part of the rainfall-runoff relationship that is most relevant to the research question is an invaluable asset. For that reason, the NSGA-II is the best option from those given here.

While there is obvious desirability for the ability to calibrate in a biased way to ensure that the relevant portion of the rainfall-runoff relationship is captured, it is worth remembering what calibration *is*. All research studies that wish to use SWMM must first establish what Knighton et al. Knighton et al. (2016) called a "feasible parameter space". That is the range of possible values that

each of the parameters might conceivable have. That is the reasonable range of parameter values by which the catchment of study must actually physically be described. Calibrating differently to optimize different parts of the hydrograph response seems reasonable, until the physical catchment is recalled. It is, in a way, nonsensical for the average slope of a catchment to be 0.01 *and* 0.05, depending on which makes the model result more attractive.

A.10.3 For Uncertainty Analysis

- Generalized likelihood uncertainty estimation - various formal Bayesian likelihood functions (Knighton et al., 2016; Sun et al., 2013; Zhang et al., 2015)
- Markov-chain Monte Carlo simulation (Muleta et al., 2013)

Despite the weirdly political atmosphere surrounding the uncertainty analysis community, the authors of the works consulted for this report agreed that this was an under-appreciated and important component of analysis undertaken with SWMM. For many studies, the level of scientific objectivity that uncertainty analysis grants is not necessary to arrive at useful conclusions. However, for truly sophisticated studies, or for research that is not constrained by the brevity of a journal article (such as a thesis or dissertation), uncertainty analysis is a requirement. Good arguments have been made on behalf of using GLUE with Bayesian likelihood functions.

A.11 Conclusion

This report consists of a representative collection of published work that has been done using SWMM, specifically, the calibration of parameters in SWMM to make

the model more useful. It is clear that there has been tremendous variability and even dissention over the best way for this to be done.

The three steps of the Rossman-defined calibration process could perhaps be expanded to four, as the process cannot begin without a firmly and reasonably set feasible parameter space. Researchers invariably have, and will continue to, develop these parameter spaces from published literature review of journal articles or hydrological handbooks. However, agreement among researchers for the standard protocol for calibration seems to end there. Establishing a SWMM-supported, comprehensive, standardized methodology for the remaining three steps of the calibration process would be a valuable addition to this highly used program.

B Compilation of Temporary Supporting Files for SWMMCALPY

B.1 Parameter_ranges.txt

Parameter_ranges.txt is a keyword-value style list of the surface parameters being calibrated by SWMMCALPY. The parameter is listed, followed by their corresponding low and high value for use in the uniform distribution sampling that creates random guesses.

	Parameter_ranges	
Percent Impervious (Low High)	- 0	100
Width (Low High)	- 1	1000
Slope (Low High)	- 0.0001	0.4
Impervious N (Low High)	- 0.001	0.1
Pervious N (Low High)	- 0.001	0.1
Impervious Storage (Low High)	- 0.001	3
Pervious Storage (Low High)	- 0.001	3
Percent Zero Storage (Low High)	- 0	100

B.2 Trial_observation.dat

Trial_observation.dat contains the synthetically created observed hydrograph for comparison against SWMM simulations. Header metadata includes the root location and the type of data that is being reported. What follows is a MM/DD/YYYY HH:MM:SS Value format for the data being reported. Values less than 0 were included to build generality into SWMMCALPYs ability to deal with missing data.

```
; Node 18 Total Inflow
18
01/01/1998 01:00:00 0
01/01/1998 02:00:00 1
01/01/1998 03:00:00 5
01/01/1998 04:00:00 10
01/01/1998 05:00:00 19
01/01/1998 06:00:00 25
01/01/1998 07:00:00 20
01/01/1998 08:00:00 12
01/01/1998 09:00:00 0
01/01/1998 10:00:00 0.06
01/01/1998 11:00:00 0.04
01/01/1998 12:00:00 0.01
01/01/1998 13:00:00 0
01/01/1998 14:00:00 0
01/01/1998 15:00:00 0
01/01/1998 16:00:00 0
01/01/1998 17:00:00 0
01/01/1998 18:00:00 -9999
01/01/1998 19:00:00 -9999
01/01/1998 20:00:00 0
01/01/1998 21:00:00 0
01/01/1998 22:00:00 0
01/01/1998 23:00:00 0
01/02/1998 00:00:00 0
01/02/1998 01:00:00 0
01/02/1998 02:00:00 0
01/02/1998 03:00:00 0
01/02/1998 04:00:00 0
01/02/1998 05:00:00 0.82
01/02/1998 06:00:00 0.6
01/02/1998 07:00:00 0.19
01/02/1998 08:00:00 0.11
01/02/1998 09:00:00 0.06
01/02/1998 10:00:00 0.04
01/02/1998 11:00:00 0.01
01/02/1998 12:00:00 0
```

B.3 Example1.inp

Example1.inp is the full input file that EPA SWMM reads to execute model runs and produce the GUI.

```
[TITLE]
;;Project Title/Notes
Example 1

[OPTIONS]
;;Option             Value
FLOW_UNITS           CFS
INFILTRATION         HORTON
FLOW_ROUTING         KINWAVE
LINK_OFFSETS         DEPTH
MIN_SLOPE            0
ALLOW_PONDING        NO
SKIP_STEADY_STATE    NO

START_DATE           01/01/1998
START_TIME           00:00:00
REPORT_START_DATE    01/01/1998
REPORT_START_TIME    00:00:00
END_DATE             01/02/1998
END_TIME             12:00:00
SWEEP_START          01/01
SWEEP_END            12/31
DRY_DAYS             5
REPORT_STEP          00:05:00
WET_STEP             00:15:00
DRY_STEP             01:00:00
ROUTING_STEP         0:01:00

INERTIAL_DAMPING     PARTIAL
NORMAL_FLOW_LIMITED  BOTH
FORCE_MAIN_EQUATION  D-W
VARIABLE_STEP        0.75
LENGTHENING_STEP     0
MIN_SURFAREA         12.557
MAX_TRIALS           8
HEAD_TOLERANCE       0.005
SYS_FLOW_TOL         5
LAT_FLOW_TOL         5
MINIMUM_STEP         0.5
THREADS              1

[EVAPORATION]
;;Data Source    Parameters
;;-------------- ----------------
CONSTANT         0.0
DRY_ONLY         NO

[RAINGAGES]
```

```
;;Name           Format    Interval SCF      Source
;;-------------- --------- ------ ------ ----------
RG1              INTENSITY 1:00     1.0      TIMESERIES TS1

[SUBCATCHMENTS]
;;Name           Rain Gage        Outlet           Area     %Imperv  Width    %Slope
  CurbLen  SnowPack
;;-------------- ---------------- ---------------- -------- -------- --------
-------- -------- ----------------
1                RG1              9                10       50       500      0.01
  0
2                RG1              10               10       50       500      0.01
  0
3                RG1              13               10       50       500      0.01
  0
4                RG1              22               10       50       500      0.01
  0
5                RG1              15               10       50       500      0.01
  0
6                RG1              23               10       10       500      0.01
  0
7                RG1              19               10       10       500      0.01
  0
8                RG1              18               10       10       500      0.01
  0

[SUBAREAS]
;;Subcatchment   N-Imperv   N-Perv     S-Imperv   S-Perv     PctZero    RouteTo
PctRouted
;;-------------- ---------- ---------- ---------- ---------- ---------- ----------
----------
1                0.001      0.10       0.05       0.05       25         OUTLET
2                0.001      0.10       0.05       0.05       25         OUTLET
3                0.001      0.10       0.05       0.05       25         OUTLET
4                0.001      0.10       0.05       0.05       25         OUTLET
5                0.001      0.10       0.05       0.05       25         OUTLET
6                0.001      0.10       0.05       0.05       25         OUTLET
7                0.001      0.10       0.05       0.05       25         OUTLET
8                0.001      0.10       0.05       0.05       25         OUTLET

[INFILTRATION]
;;Subcatchment   MaxRate    MinRate    Decay      DryTime    MaxInfil
;;-------------- ---------- ---------- ---------- ---------- ----------
1                0.7        0.3        4.14       0.50       0
2                0.7        0.3        4.14       0.50       0
3                0.7        0.3        4.14       0.50       0
4                0.7        0.3        4.14       0.50       0
5                0.7        0.3        4.14       0.50       0
```

```
6               0.7         0.3          4.14        0.50        0
7               0.7         0.3          4.14        0.50        0
8               0.7         0.3          4.14        0.50        0

[JUNCTIONS]
;;Name           Elevation   MaxDepth    InitDepth   SurDepth    Aponded
;;-------------- ----------- ---------- ---------- ---------- ----------
9               1000        3           0           0           0
10              995         3           0           0           0
13              995         3           0           0           0
14              990         3           0           0           0
15              987         3           0           0           0
16              985         3           0           0           0
17              980         3           0           0           0
19              1010        3           0           0           0
20              1005        3           0           0           0
21              990         3           0           0           0
22              987         3           0           0           0
23              990         3           0           0           0
24              984         3           0           0           0

[OUTFALLS]
;;Name           Elevation   Type        Stage Data       Gated    Route To
;;-------------- ----------- ---------- ---------------- -------- ----------------
18              975         FREE                         NO

[CONDUITS]
;;Name           From Node        To Node          Length     Roughness   InOffset
OutOffset   InitFlow    MaxFlow
;;-------------- ---------------- ---------------- ---------- ---------- ----------
---------- ---------- ----------
1               9                10               400        0.01        0
0           0           0
4               19               20               200        0.01        0
0           0           0
5               20               21               200        0.01        0
0           0           0
6               10               21               400        0.01        0
1           0           0
7               21               22               300        0.01        1
1           0           0
8               22               16               300        0.01        0
0           0           0
10              17               18               400        0.01        0
0           0           0
11              13               14               400        0.01        0
0           0           0
12              14               15               400        0.01        0
```

```
0          0          0
13               15               16               400        0.01       0
0          0          0
14               23               24               400        0.01       0
0          0          0
15               16               24               100        0.01       0
0          0          0
16               24               17               400        0.01       0
0          0          0

[XSECTIONS]
;;Link           Shape        Geom1            Geom2      Geom3      Geom4
Barrels    Culvert
;;-------------- ------------ ---------------- ---------- ---------- ----------
---------- ----------
1                CIRCULAR     1.5              0          0          0          1

4                CIRCULAR     1                0          0          0          1

5                CIRCULAR     1                0          0          0          1

6                CIRCULAR     1                0          0          0          1

7                CIRCULAR     2                0          0          0          1

8                CIRCULAR     2                0          0          0          1

10               CIRCULAR     2                0          0          0          1

11               CIRCULAR     1.5              0          0          0          1

12               CIRCULAR     1.5              0          0          0          1

13               CIRCULAR     1.5              0          0          0          1

14               CIRCULAR     1                0          0          0          1

15               CIRCULAR     2                0          0          0          1

16               CIRCULAR     2                0          0          0          1


[POLLUTANTS]
;;Name           Units  Crain      Cgw        Crdii      Kdecay     SnowOnly
Co-Pollutant     Co-Frac    Cdwf       Cinit
;;-------------- ------ ---------- ---------- ---------- ---------- ----------
---------------- ---------- ---------- ----------
TSS              MG/L   0.0        0.0        0          0.0        NO         *
```

```
          0.0        0          0
Lead              UG/L   0.0        0.0        0          0.0        NO         TSS
          0.2        0          0

[LANDUSES]
;;               Sweeping   Fraction   Last
;;Name           Interval   Available  Swept
;;-------------- ---------- ---------- ----------
Residential
Undeveloped

[COVERAGES]
;;Subcatchment   Land Use         Percent
;;-------------- ---------------- ----------
1                Residential      100.00
2                Residential      50.00
2                Undeveloped      50.00
3                Residential      100.00
4                Residential      50.00
4                Undeveloped      50.00
5                Residential      100.00
6                Undeveloped      100.00
7                Undeveloped      100.00
8                Undeveloped      100.00

[LOADINGS]
;;Subcatchment   Pollutant        Buildup
;;-------------- ---------------- ----------

[BUILDUP]
;;Land Use       Pollutant        Function   Coeff1     Coeff2     Coeff3     Per
Unit
;;-------------- ---------------- ---------- ---------- ---------- ----------
----------
Residential      TSS              SAT        50         0          2          AREA

Residential      Lead             NONE       0          0          0          AREA

Undeveloped      TSS              SAT        100        0          3          AREA

Undeveloped      Lead             NONE       0          0          0          AREA


[WASHOFF]
;;Land Use       Pollutant        Function   Coeff1     Coeff2     SweepRmvl
BmpRmvl
;;-------------- ---------------- ---------- ---------- ---------- ----------
----------
```

```
Residential      TSS             EXP        0.1        1          0          0

Residential      Lead            EMC        0          0          0          0

Undeveloped      TSS             EXP        0.1        0.7        0          0

Undeveloped      Lead            EMC        0          0          0          0


[TIMESERIES]
;;Name           Date       Time       Value
;;-------------- ---------- ---------- ----------
;RAINFALL
TS1                         0:00       0.0
TS1                         1:00       0.25
TS1                         2:00       0.5
TS1                         3:00       0.8
TS1                         4:00       0.4
TS1                         5:00       0.1
TS1                         6:00       0.0
TS1                         27:00      0.0
TS1                         28:00      0.4
TS1                         29:00      0.2
TS1                         30:00      0.0

[REPORT]
;;Reporting Options
INPUT      NO
CONTROLS   NO
SUBCATCHMENTS ALL
NODES ALL
LINKS ALL

[TAGS]

[MAP]
DIMENSIONS 0.000 0.000 10000.000 10000.000
Units      None

[COORDINATES]
;;Node           X-Coord            Y-Coord
;;-------------- ------------------ ------------------
9                4042.110           9600.000
10               4105.260           6947.370
13               2336.840           4357.890
14               3157.890           4294.740
15               3221.050           3242.110
16               4821.050           3326.320
```

```
17              6252.630          2147.370
19              7768.420          6736.840
20              5957.890          6589.470
21              4926.320          6105.260
22              4421.050          4715.790
23              6484.210          3978.950
24              5389.470          3031.580
18              6631.580          505.260

[VERTICES]
;;Link          X-Coord           Y-Coord
;;------------- ----------------- -----------------
10              6673.680          1368.420

[Polygons]
;;Subcatchment  X-Coord           Y-Coord
;;------------- ----------------- -----------------
1               3936.840          6905.260
1               3494.740          6252.630
1               273.680           6336.840
1               252.630           8526.320
1               463.160           9200.000
1               1157.890          9726.320
1               4000.000          9705.260
2               7600.000          9663.160
2               7705.260          6736.840
2               5915.790          6694.740
2               4926.320          6294.740
2               4189.470          7200.000
2               4126.320          9621.050
3               2357.890          6021.050
3               2400.000          4336.840
3               3031.580          4252.630
3               2989.470          3389.470
3               315.790           3410.530
3               294.740           6000.000
4               3473.680          6105.260
4               3915.790          6421.050
4               4168.420          6694.740
4               4463.160          6463.160
4               4821.050          6063.160
4               4400.000          5263.160
4               4357.890          4442.110
4               4547.370          3705.260
4               4000.000          3431.580
4               3326.320          3368.420
4               3242.110          3536.840
4               3136.840          5157.890
```

4	2589.470	5178.950
4	2589.470	6063.160
4	3284.210	6063.160
4	3705.260	6231.580
4	4126.320	6715.790
5	2568.420	3200.000
5	4905.260	3136.840
5	5221.050	2842.110
5	5747.370	2421.050
5	6463.160	1578.950
5	6610.530	968.420
5	6589.470	505.260
5	1305.260	484.210
5	968.420	336.840
5	315.790	778.950
5	315.790	3115.790
6	9052.630	4147.370
6	7894.740	4189.470
6	6442.110	4105.260
6	5915.790	3642.110
6	5326.320	3221.050
6	4631.580	4231.580
6	4568.420	5010.530
6	4884.210	5768.420
6	5368.420	6294.740
6	6042.110	6568.420
6	8968.420	6526.320
7	8736.840	9642.110
7	9010.530	9389.470
7	9010.530	8631.580
7	9052.630	6778.950
7	7789.470	6800.000
7	7726.320	9642.110
8	9073.680	2063.160
8	9052.630	778.950
8	8505.260	336.840
8	7431.580	315.790
8	7410.530	484.210
8	6842.110	505.260
8	6842.110	589.470
8	6821.050	1178.950
8	6547.370	1831.580
8	6147.370	2378.950
8	5600.000	3073.680
8	6589.470	3894.740
8	8863.160	3978.950

[SYMBOLS]

;;Gage	X-Coord	Y-Coord
RG1	10084.210	8210.530

C Compilation of Commented Code Comprising SWMMCALPY

Figure C.1 is an illustration of how the five scripts that comprise SWMMCALPY communicate with each other.

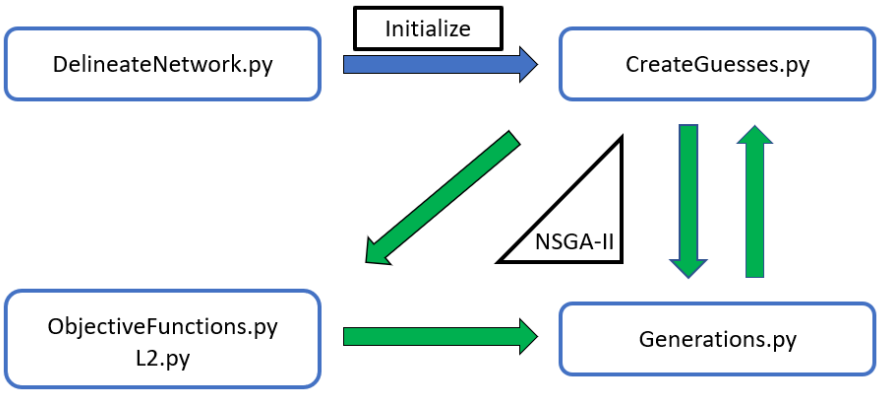


Figure C.1: Schematic of the communication patterns between the five scripts comprising SWMMCALPY

DelineateNetwork.py is passed to CreateGuesses.py once to initialize the scope of the calibration problem. Subsequently, CreateGuesses.py passes a generation of input files to Objective_functions.py and L2.py to calculate the objective functions, which are assessed and managed by the Generations.py script. Generations.py then passes enlists CreateGuesses.py to generate the next generation to be evaluated, sorted, and culled/persisted. Objective_functions.py and L2.py are grouped because, while being separate scripts, they share the same overall purpose of calculating objective function values for individual guesses.

C.1 DelineateNetwork.py

DelineateNetwork.py shows how SWMMCALPY leverages PySWMM and NetworkX to determine the clipped subsystem for a given calibration problem.

DelineateNetwork

```
#The primary purpose of DelineateNetwork.py is to use PySWMM and NetworkX to define
the scope of the calibration problem
# within a given SWMM system

#Classes from PySWMM and NetworkX
from pyswmm import Simulation, Nodes, Links, Subcatchments
import networkx as nx

#Name of the SWMM input file starting point and the name of the root location
inputfilename = 'Example1.inp'
root = '18'

#Initialize global variables
global network, subnetwork

#network is the entire SWMM system
network = nx.DiGraph()

#subnetwork is just the components upstream of root
subnetwork = nx.DiGraph()

#Create network
def createnetwork(inputfilename):
    global sim
    linklist = []

    #Allows the PySWMM wrapper to access the input file
    sim = Simulation(inputfilename)

    #Read each subcatchment and add the node to network
    for sub in Subcatchments(sim):
        network.add_node(sub.subcatchmentid)

    #Read each node (as defined by PySWMM) and add the node (as defined by NetworkX)
to network
    for nodes in Nodes(sim):
        network.add_node(nodes.nodeid)

    #Read each link and add the edge to network
    for link in Links(sim):
        linklist.append(link.connections)
    network.add_edges_from(linklist)

    #Read the subcatchment connections and add the edge to network
    for sub in Subcatchments(sim):
        network.add_edge(sub.subcatchmentid,sub.connection[1])
    return
```

```
#Create subnetwork
def subnetworkdelineation(root):

    #Begin by populating subnetwork with the root node
    subnetwork.add_node(root)

    #Determine the list of upstream nodes
    predecessorlist = list(network.predecessors(root))

    #If there are no predecessors, return
    if predecessorlist == []:
        return
    else:

        #Recursive search through predecessors, add new nodes to subnetwork
        for i in predecessorlist:
            subnetwork.add_node(i)
            subnetworkdelineation(i)

    #Global subnetwork is only nodes, no edges
    return

#Determine which subnetwork nodes are subcatchments
def subnetwork_subcatchments(inputfilename, root):

    #Initialize subcatchment list, faster to define as global rather than return
    global list_of_subcatchments
    list_of_subcatchments = []

    #Create network
    createnetwork(inputfilename)

    #Create subnetwork
    subnetworkdelineation(root)

    #Compare subnetwork to PySWMM subcatchments, if there's a match, at it to
list_of_subcatchments
    for subcatchment in Subcatchments(sim):
        subcatchmentname = subcatchment.subcatchmentid

        for subnode in subnetwork:
            if subnode == subcatchmentname:
                list_of_subcatchments.append(subcatchmentname)
    return

#Calls function that defines list_of_subcatchments variable
# runs automatically when DelineateNetwork is imported
subnetwork_subcatchments(inputfilename, root)
```

C.2 CreateGuesses.py

CreateGuesses.py contains the portions of SWMMCALPY that pertain to parsing a SWMM input file (like the one shown in Chapter B), manipulating the parameter information held within, and inserting the manipulated parameter set information into a new input file. CreateGuesses.py references the clipped subsystem determined by DelineateNetwork.py

```
#Random package used to select from uniform distribution
from random import *

#Import the list_of_subcatchments variable from DelineateNetwork
import DelineateNetwork_Commented
import numpy as np

#Defines feasible parameter space
constraintfilename = 'Parameter_ranges.txt'
inputfilename = 'Example1.inp'
root = '18'

#List of "trialfileXX" is used to define a generation of NSGA-II guesses
filelist = ['trialfile01.inp', 'trialfile02.inp', 'trialfile03.inp',
'trialfile04.inp', 'trialfile05.inp',
            'trialfile06.inp', 'trialfile07.inp',
            'trialfile08.inp', 'trialfile09.inp', 'trialfile10.inp',
'trialfile11.inp', 'trialfile12.inp',
            'trialfile13.inp', 'trialfile14.inp',
            'trialfile15.inp', 'trialfile16.inp', 'trialfile17.inp',
'trialfile18.inp', 'trialfile19.inp',
            'trialfile20.inp', 'trialfile21.inp',
            'trialfile22.inp', 'trialfile23.inp', 'trialfile24.inp',
'trialfile25.inp', 'trialfile26.inp',
            'trialfile27.inp', 'trialfile28.inp',
            'trialfile29.inp', 'trialfile30.inp', 'trialfile31.inp',
'trialfile32.inp', 'trialfile33.inp',
            'trialfile34.inp', 'trialfile35.inp',
            'trialfile36.inp', 'trialfile37.inp', 'trialfile38.inp',
'trialfile39.inp', 'trialfile40.inp',
            'trialfile41.inp', 'trialfile42.inp',
            'trialfile43.inp', 'trialfile44.inp', 'trialfile45.inp',
'trialfile46.inp', 'trialfile47.inp',
            'trialfile48.inp', 'trialfile49.inp',
            'trialfile50.inp', 'trialfile51.inp', 'trialfile52.inp',
'trialfile53.inp', 'trialfile54.inp',
            'trialfile55.inp', 'trialfile56.inp',
            'trialfile57.inp', 'trialfile58.inp', 'trialfile59.inp',
'trialfile60.inp', 'trialfile61.inp',
            'trialfile62.inp', 'trialfile63.inp',
            'trialfile64.inp', 'trialfile65.inp', 'trialfile66.inp',
'trialfile67.inp', 'trialfile68.inp',
            'trialfile69.inp', 'trialfile70.inp',
            'trialfile71.inp', 'trialfile72.inp', 'trialfile73.inp',
'trialfile74.inp', 'trialfile75.inp',
            'trialfile76.inp', 'trialfile77.inp',
            'trialfile78.inp', 'trialfile79.inp', 'trialfile80.inp',
'trialfile81.inp', 'trialfile82.inp',
```

```
            'trialfile83.inp', 'trialfile84.inp',
            'trialfile85.inp', 'trialfile86.inp', 'trialfile87.inp',
'trialfile88.inp', 'trialfile89.inp',
            'trialfile90.inp', 'trialfile91.inp',
            'trialfile92.inp', 'trialfile93.inp', 'trialfile94.inp',
'trialfile95.inp', 'trialfile96.inp',
            'trialfile97.inp', 'trialfile98.inp',
            'trialfile99.inp', 'trialfile100.inp']

#Reads the feasible parameter space into global variables
def readparametersfromfile(constraintfilename):
    global percentimpervious, width, slope, impervious_n, pervious_n,
impervious_storage, pervious_storage, percent_zero_storage
    constraintfile = open(constraintfilename, 'r')

    #Reads each line in constraintfile, preserves generality to change order or
include/exclude parameters
    # Assumes file is organized as Keyword - High Low
    for line in constraintfile:
        if (line.find("Percent Impervious") != -1):
            percentimpervious = []
            templine = line.split()
            #Add High and Low parameter limits to global list variable
            percentimpervious.append(float(templine[-2]))
            percentimpervious.append(float(templine[-1]))
        elif (line.find("Width") != -1):
            width = []
            templine = line.split()
            width.append(float(templine[-2]))
            width.append(float(templine[-1]))
        elif (line.find("Slope") != -1):
            slope = []
            templine = line.split()
            slope.append(float(templine[-2]))
            slope.append(float(templine[-1]))
        elif (line.find("Impervious N") != -1):
            impervious_n = []
            templine = line.split()
            impervious_n.append(float(templine[-2]))
            impervious_n.append(float(templine[-1]))
        elif (line.find("Pervious N") != -1):
            pervious_n = []
            templine = line.split()
            pervious_n.append(float(templine[-2]))
            pervious_n.append(float(templine[-1]))
        elif (line.find("Impervious Storage") != -1):
            impervious_storage = []
            templine = line.split()
```

```
            impervious_storage.append(float(templine[-2]))
            impervious_storage.append(float(templine[-1]))
        elif (line.find("Pervious Storage") != -1):
            pervious_storage = []
            templine = line.split()
            pervious_storage.append(float(templine[-2]))
            pervious_storage.append(float(templine[-1]))
        elif (line.find("Percent Zero Storage") != -1):
            percent_zero_storage = []
            templine = line.split()
            percent_zero_storage.append(float(templine[-2]))
            percent_zero_storage.append(float(templine[-1]))
    return

#Call function as standalone, only needs to occur once
readparametersfromfile(constraintfilename)

#Counts the number of lines in input file
# Used in later for loops because while loops were causing issues
def countsubcatchments(inputfilename):
    global count
    with open(inputfilename) as swmmput:
        count = 0
        for line in swmmput:
            count += 1
        return

#Call function as standalone, only needs to occur once
countsubcatchments(inputfilename)

#Reads the surface parameters from the initial input file into a 2D list
def read_initial_parameters(inputfilename):

    #SWMM surface parameters are stored under [SUBCATCHMENTS] and [SUBAREAS] headers
in .inp file
    subc_params = []
    subarea_params = []

    #subcatchment_parameters is the name of the 2D list of parameters
    # subc_names is compared to list_of_subcatchments from DelineateNetwork.py
    global subcatchment_parameters, subc_names
    subc_names = []
    subcatchment_parameters = []

    #Open the SWMM inputfile as read only
    inputfile = open(inputfilename, 'r')

    #For each line in the inputfile
```

```
    for line in inputfile:

        #Find where the [SUBCATCHMENTS] header begins
        if (line.find("[SUBCATCHMENTS]") != -1):
            line = inputfile.readline()

            #Use count to define how long the parser should keep looking for
information
            for i in range(count):
                templine = list(line)

                #Check if the line contains metadata
                if templine[0] == ";" or templine[0] == " " or len(templine) < 10:
                    line = inputfile.readline()
                    continue

                #Check if the next header has been reached
                elif (line.find("[") != -1):
                    break

                #Read in the parameter as a split line
                else:
                    linesplit = line.split()

                    #[SUBCATCHMENT] area parameters are in token locations 4-7
                    subc_params.append(linesplit[4:7])

                    #The subcatchment name is in token location 0
                    subc_names.append(linesplit[0])
                    line = inputfile.readline()

        #Same search within the [SUBAREAS] header
        if (line.find("[SUBAREAS]") != -1):
            line = inputfile.readline()
            for i in range(count):
                templine = list(line)
                if templine[0] == ";" or templine[0] == " " or len(templine) < 10:
                    line = inputfile.readline()
                    continue
                elif (line.find("[") != -1):
                    break
                else:
                    linesplit = line.split()
                    subarea_params.append(linesplit[1:6])
                    line = inputfile.readline()

    #Compiles the 2D subc_params and subarea_params lists into a single 2D list,
subcatchment parameters, that contains
```

```
    # the surface parameters for each subcatchment in a 1D list as a 2D list
element.
    for i in range(len(subc_params)):
        for j in range(len(subarea_params[i])):
            subc_params[i].append(subarea_params[i][j])
        subcatchment_parameters.append(subc_params[i])
    return

#Only called once at the beginning of SWMMCALPY to get a starting location of the
parameter set
read_initial_parameters(inputfilename)

#Transforms a 2D list into a 1D list for
def transformation_flatten(twoDlistinput):
    oneDlistoutput = []

    #For every 1D list element in the 2D list, add it to the 1D list.  This
preserves order but destroys the 2nd Dim.
    for i in range(len(twoDlistinput)):
        for j in range(len(twoDlistinput[i])):
            oneDlistoutput.append(twoDlistinput[i][j])
    return(oneDlistoutput)

#Clips the initial guess down to only the clipped subsystem being scrutinized
def compile_initial_guess():
    global initial_guess_flat, relevant_subcatchment_parameters
    relevant_subcatchment_indices = []

    #Search for the subcatchment names from read_initial_parameters() and
DelineateNetwork.list_of_subcatchments
    for allsub in subc_names:
        for upstreamsub in DelineateNetwork_Commented.list_of_subcatchments:

            #If they match, add the index to a dummy list
            if allsub == upstreamsub:
                relevant_subcatchment_indices.append(subc_names.index(allsub))
    relevant_subcatchment_parameters = []

    #For each index in the dummy list, grab the corresponding element in the 2D
subcatchment_parameters list
    for i in relevant_subcatchment_indices:
        relevant_subcatchment_parameters.append(subcatchment_parameters[i])

    #Transform the clipped 2D list into the 1D list for manipulation
    initial_guess_flat = transformation_flatten(relevant_subcatchment_parameters)
    return

#Only needs to happen once to determine the location of the relevant parameter set
```

```
compile_initial_guess()

#Another transformation function that casts the strings from the parsed list into
floats
def caststringsasfloats():
    initial_guess_floats = []

    #For every value in a 1D list, recast the value as a float and add it to a new
float list
    for string_value in initial_guess_flat:
        initial_guess_floats.append(float(string_value))
    return(initial_guess_floats)

#Function that generates a mutated guess from some parent guess
def createrandomsetofP():

    #Initial transformation of a parsed set of guesses into floats
    temporaryguess = caststringsasfloats()

    #For each parameter in the list of floats
    for parameter in range(len(temporaryguess)):
        binary_setter = uniform(0, 1)

        # Basically flip a coin
        if binary_setter < 0.5:
            # if heads, go to the next parameter

            continue
        else:
            # if tails,

            if parameter % 8 == 0:
                # AND the parameter index is divisible by 8 (the number of
calibratable parameters in this first cut)

                temporaryguess[parameter] = uniform(percentimpervious[0],
percentimpervious[1])
                # Reassign that parameter in the list of floats to a random number
generated from a uniform distribution
                # whose boundaries were specified in the "readparametersfromfile"
parser function

            elif parameter % 8 == 1:
                # Do the same for width, slope, etc

                temporaryguess[parameter] = uniform(width[0], width[1])
            elif parameter % 8 == 2:
                temporaryguess[parameter] = uniform(slope[0], slope[1])
```

```
            elif parameter % 8 == 3:
                temporaryguess[parameter] = uniform(impervious_n[0],
impervious_n[1])
            elif parameter % 8 == 4:
                temporaryguess[parameter] = uniform(pervious_n[0], pervious_n[1])
            elif parameter % 8 == 5:
                temporaryguess[parameter] = uniform(impervious_storage[0],
impervious_storage[1])
            elif parameter % 8 == 6:
                temporaryguess[parameter] = uniform(pervious_storage[0],
pervious_storage[1])
            elif parameter % 8 == 7:
                temporaryguess[parameter] = uniform(percent_zero_storage[0],
percent_zero_storage[1])

    #Retuns a mutated guess
    return (temporaryguess)

#Transformation from floats back into strings for reinsertion into input file
def castfloatsasstrings():

    #Calls the random guess generator to initialize the list to be recast
    floattostring = createrandomsetofP()
    guess_strings = []

    #Add the guess back element wise to a dummy list of strings
    for float in floattostring:
        guess_strings.append(str(float))
    return(guess_strings)

#Inverse of the flatten transformation, reexpands a 1D list into a 2D array where
each element in a row is one of
# each subcatchment's surface parameters
def transformation_fatten(oneDlistinput):

    #Initialize a 2D np array of zeros to be repopulated, number of columns is the
number of subcatchments, and
    # the number of rows is the number of surface parameters
    new_twoDlistoutput = np.zeros((len(relevant_subcatchment_parameters[0]),
len(relevant_subcatchment_parameters)))
    row_count = -1
    col_count = 0

    #For each parameter in the 1D list
    for oneDparameter in oneDlistinput:
        row_count = row_count + 1

        #Populate the array row by row, essentially each subcatchment gets populated
```

```
top to bottom
        if row_count < len(relevant_subcatchment_parameters[0]):

            # If the row count is still less than the number of different parameters
(i.e. < 8)
            new_twoDlistoutput[row_count][col_count] = oneDparameter
        else:

            # Once the number of rows in each column has been exceeded, move to the
next column and restart row count
            row_count = 0
            col_count = col_count + 1
            new_twoDlistoutput[row_count][col_count] = oneDparameter
    return (new_twoDlistoutput)

#Reinsert the np array of strings into the input file
def insertguessestoinputfile(inputfile, trialfile):

    #Initialize the np array, guess, as a recast version of a randomly generated
guess
    guess = transformation_fatten(castfloatsasstrings())

    #Open and read the initial input file and store the information in the contents
variable
    with open(inputfile, 'r') as swmmput:
        contents = swmmput.readlines()

        #Go back to the top of the inputfile
        swmmput.seek(0)

        #Read input file again and where the parameters are found insert the guess
parameter into the contents variable
        for line in swmmput:
            if line.find('[SUBCATCHMENTS]') != -1:
                for i in range(count):
                    line = swmmput.readline()
                    linelist = list(line)
                    if linelist[0] == " " or linelist[0] == ";" or len(linelist) <
10:
                        continue
                    elif (line.find('[SUBAREAS]') != -1):
                        break
                    else:

                        #When a subcatchment name in the inputfile is found, check
if that subcatchment is in the
                        # clipped subsystem
                        for sub in DelineateNetwork_Commented.list_of_subcatchments:
```

```
                            templine = contents.index(line)
                            splitline = contents[templine].split()

                            #If the subcatchment is relevant, add the guess
parameter to the contents variable
                            if splitline[0] == sub:
                                splitline[4] =
str(guess[0][DelineateNetwork_Commented.list_of_subcatchments.index(sub)])
                                splitline[5] =
str(guess[1][DelineateNetwork_Commented.list_of_subcatchments.index(sub)])
                                splitline[6] =
str(guess[2][DelineateNetwork_Commented.list_of_subcatchments.index(sub)])

                                #Add some spaces to the input file to improve
readability
                                contents[templine] = "      ".join(splitline) + "\n"
                                break
            if line.find('[SUBAREAS]') != -1:

                # This is the same thing, but for the [SUBAREAS] header which houses
5 of the 8 parameters
                for i in range(count):
                    line = swmmput.readline()
                    linelist = list(line)
                    if linelist[0] == " " or linelist[0] == ";" or len(linelist) <
10:
                        continue
                    elif (line.find('[') != -1):
                        break
                    else:
                        for sub in DelineateNetwork_Commented.list_of_subcatchments:
                            templine = contents.index(line)
                            splitline = contents[templine].split()
                            if splitline[0] == sub:
                                splitline[1] =
str(guess[3][DelineateNetwork_Commented.list_of_subcatchments.index(sub)])
                                splitline[2] =
str(guess[4][DelineateNetwork_Commented.list_of_subcatchments.index(sub)])
                                splitline[3] =
str(guess[5][DelineateNetwork_Commented.list_of_subcatchments.index(sub)])
                                splitline[4] =
str(guess[6][DelineateNetwork_Commented.list_of_subcatchments.index(sub)])
                                splitline[5] =
str(guess[7][DelineateNetwork_Commented.list_of_subcatchments.index(sub)])
                                contents[templine] = "      ".join(splitline) + '\n'
                                break

    #Close the original swmm input file
```

```
    swmmput.close()

    #Open the trial file and write all the lines from the contents variable
    with open(trialfile, 'w') as newfile:
        for i in range(count):
            newfile.write(contents[i])

    #Then close the trialfile
    newfile.close()
    return

#Iterate the process of generating a random guess and creating a new input file 100
times to create a generation
def create_generation(inputfilename, filelist):
    for trialfile in filelist:
        insertguessestoinputfile(inputfilename, trialfile)
    return

#Create the first generation of guesses.  Result is 100 new input files in the
directory.
create_generation(inputfilename, filelist)
```

C.3 L2.py

L2.py is a script that calculates the NED objective function by comparing the parameter set obtained from a guess input file to the one obtained from the original input file. The NED objective function is calculated in a separate script than the other objective functions because it operates within the "feasible parameter space" rather than the "feasible performance space" like the other three objective functions. L2.py references CreateGuesses.py to access functions that grab the two parameter sets.

```
#Import CreateGuesses to access functions
import CreateGuesses

#Import math perform squared operations
import math

#Collect the list of parameter values from each guess
def get_random_guess(trialfilename):

    #Initialize a variable containing the string list of parameters for each guess
    global random_guess_str

    #Determine this list of strings by calling the return of the
transformation_flatten function, with the 2D list
    # from the read_initial_parameters function passed as an argument.
    random_guess_str =
CreateGuesses.transformation_flatten(CreateGuesses.read_initial_parameters(trialfile
name))
    return random_guess_str

#Determine the NED between the guess and the original inputfile
def L2norm(trialfilename):

    #Collect the string list of parameter values from each guess
    get_random_guess(trialfilename)

    #Collect the float list of parameter values from the original estimate
    initial_guess = CreateGuesses.caststringsasfloats()

    #Initialize lists and variables to be populated
    random_guess = []
    numerator = []
    denominator = []
    num_sum = 0
    denom_sum = 0
    L2 = 0

    #Cast the string list of parameter values from each guess into a float list
    for i in random_guess_str:
        random_guess.append(float(i))

    #Compare the two lists of parameters, append the differences squared to a dummy
list
    for parameter in initial_guess:
        num = (parameter - random_guess[initial_guess.index(parameter)])
        num_squared = math.pow(num, 2)
        numerator.append(num_squared)
        denom = parameter + random_guess[initial_guess.index(parameter)]
```

```
        denom_squared = math.pow(denom, 2)
        denominator.append(denom_squared)

    #Sum the dummy list of numerator and denominator comparisons between the two
float lists
    for j_index in numerator:
        num_sum = num_sum + j_index
        denom_sum = denom_sum + denominator[numerator.index(j_index)]

    #Compute the NED by taking the square root of the list sums
    L2 = math.sqrt(num_sum/denom_sum)

    #Return the NED float value for use in Objective_function.py
    return(L2)
```

C.4 Objective_functions.py

Objective_functions.py calculates the remaining objective functions: the NPE, NVE, and NSE_m. It does so by comparing the simulated hydrograph managed by PySWMM with the synthetically produced observed hydrograph shown in B. Objective_functions.py also calculates the aggregate functions and ranks the guesses within a generation based on the aggregate function values. Objective_functions.py references L2.py to collect the values of the NED objective function for a given guess.

```
#Import pyswmm for use in accessing the swmm simulations
from pyswmm import Simulation, Nodes, Links, Subcatchments

#Import datetime to manage the time delta between recorded data
import datetime

#Import re so that lines can be split on more than one criteria
import re

#Import L2 so that all objective functions can be aggregated
import L2

#Specify the observation file for comparison to simulations.  Observations must be
equally spaced in time.
observationdatafile = "trial_observation.dat"

#Specify the location in the system of the observation data
root = "18"

#List of trial files for a given generation
filelist = ['trialfile01.inp', 'trialfile02.inp', 'trialfile03.inp',
'trialfile04.inp', 'trialfile05.inp',
            'trialfile06.inp', 'trialfile07.inp',
            'trialfile08.inp', 'trialfile09.inp', 'trialfile10.inp',
'trialfile11.inp', 'trialfile12.inp',
            'trialfile13.inp', 'trialfile14.inp',
            'trialfile15.inp', 'trialfile16.inp', 'trialfile17.inp',
'trialfile18.inp', 'trialfile19.inp',
            'trialfile20.inp', 'trialfile21.inp',
            'trialfile22.inp', 'trialfile23.inp', 'trialfile24.inp',
'trialfile25.inp', 'trialfile26.inp',
            'trialfile27.inp', 'trialfile28.inp',
            'trialfile29.inp', 'trialfile30.inp', 'trialfile31.inp',
'trialfile32.inp', 'trialfile33.inp',
            'trialfile34.inp', 'trialfile35.inp',
            'trialfile36.inp', 'trialfile37.inp', 'trialfile38.inp',
'trialfile39.inp', 'trialfile40.inp',
            'trialfile41.inp', 'trialfile42.inp',
            'trialfile43.inp', 'trialfile44.inp', 'trialfile45.inp',
'trialfile46.inp', 'trialfile47.inp',
            'trialfile48.inp', 'trialfile49.inp',
            'trialfile50.inp', 'trialfile51.inp', 'trialfile52.inp',
'trialfile53.inp', 'trialfile54.inp',
            'trialfile55.inp', 'trialfile56.inp',
            'trialfile57.inp', 'trialfile58.inp', 'trialfile59.inp',
'trialfile60.inp', 'trialfile61.inp',
            'trialfile62.inp', 'trialfile63.inp',
            'trialfile64.inp', 'trialfile65.inp', 'trialfile66.inp',
```

```
'trialfile67.inp', 'trialfile68.inp',
            'trialfile69.inp', 'trialfile70.inp',
            'trialfile71.inp', 'trialfile72.inp', 'trialfile73.inp',
'trialfile74.inp', 'trialfile75.inp',
            'trialfile76.inp', 'trialfile77.inp',
            'trialfile78.inp', 'trialfile79.inp', 'trialfile80.inp',
'trialfile81.inp', 'trialfile82.inp',
            'trialfile83.inp', 'trialfile84.inp',
            'trialfile85.inp', 'trialfile86.inp', 'trialfile87.inp',
'trialfile88.inp', 'trialfile89.inp',
            'trialfile90.inp', 'trialfile91.inp',
            'trialfile92.inp', 'trialfile93.inp', 'trialfile94.inp',
'trialfile95.inp', 'trialfile96.inp',
            'trialfile97.inp', 'trialfile98.inp',
            'trialfile99.inp', 'trialfile100.inp']

#Specify the weights for each objective function
weights = [0.25, 0.25, 0.25, 0.25]

#Read the observation data into a time series
def readobservationfile(observationdatafile):

    #Open the file as read only, place all information into a contents variable
    with open(observationdatafile, 'r') as obs_file:
        global contents
        contents = obs_file.readlines()

        #Initialize variables that will be used to compute objective functions
        global obs_data, time_difference, obs_time
        obs_data = []
        obs_time = []

        #For each line in contents variable
        for line in contents:

            #Split the line into characters
            linelist = list(line)

            #Determine if the line is metadata
            if linelist[0] == ';' or linelist[0] == ' ' or len(list(line)) < 15:
                continue

            #if not metadata
            else:
                templine = line.split()

                #If the data is a placeholder, replace it with 0
                if float(templine[-1]) < 0:
```

```
                    obs_data.append(0)

                #If the data is a true value, add it to the obs_data list
                else:
                    obs_data.append(float(templine[-1]))

                #Process the datatime data in the observation data file
                # formatting requisite is MM/DD/YYYY HH:MM:SS
                day_templine_preprocessing = line.replace(' ', ';')
                day_templine = re.split('[/|;|:]', day_templine_preprocessing)
                month = int(day_templine[0])
                day = int(day_templine[1])
                year = int(day_templine[2])
                hour = int(day_templine[3])
                minute = int(day_templine[4])
                second = int(day_templine[5])
                obs_time.append(datetime.datetime(year, month, day, hour, minute,
second))

        #Determine the time spacing of the series
        time_difference = obs_time[1] - obs_time[0]
    return

#This function only needs to be called once as the observational data doesn't change
readobservationfile(observationdatafile)

#Formulation of NPE
def normalizedpeakerror():

    #Maximum value of simulated hydrograph from PySWMM
    peak_simulation = max(hydrograph)

    #Maximum value of observed hydrograph from readobservationfile()
    peak_observation = max(obs_data)

    #Calculate NPE
    peak_error = abs(peak_simulation - peak_observation)/(peak_observation +
peak_simulation)

    #Returns the float value of NPE
    return(peak_error)

#Formulation of NVE
def normalizedvolumeerror():

    #Initivalize volume of simulation and observation hydrographs
    volume_simulation = 0
    volume_observation = 0
```

```
    #Trapezoidal approximations of hydrographs
    for sim_index in range(1,len(hydrograph)):
        volume_simulation_trapezoid =
(hydrograph[sim_index-1]+hydrograph[sim_index])*simulation_timestep/2
        volume_simulation = volume_simulation + volume_simulation_trapezoid
    for data_index in range(1,len(obs_data)):
        volume_observation_trapezoid =
(obs_data[data_index-1]+obs_data[data_index])*time_difference.total_seconds()/2
        volume_observation = volume_observation + volume_observation_trapezoid

    #Calculate NVE
    volume_error = abs(volume_simulation-volume_observation)/(volume_simulation +
volume_observation)

    #Returns the float value of NVE
    return(volume_error)

#Formulation of NSEm
def nashsutcliffe():

    #Compute the average of the observed data set
    average_obs = sum(obs_data)/len(obs_data)

    #Initialize the sum of the differences
    sum_sim_obs = 0
    sum_obs_obsave = 0

    #Bound the computation to time instances where both time series exist
    for i in range(len(min(obs_data, hydrograph))-1):

        #Update the sum of the differences for each time step
        diff_sim_obs = (obs_data[i] - hydrograph[i])**2
        sum_sim_obs = sum_sim_obs + diff_sim_obs
        diff_obs_obsave = (obs_data[i] - average_obs)**2
        sum_obs_obsave = sum_obs_obsave + diff_obs_obsave

    #Compute the final ratio for NSEm
    mNSE = sum_sim_obs/sum_obs_obsave

    #Returns the float value of NSEm
    return(mNSE)

#Compile a list of the objective function values for each guess
def objectivefunctions(filelist, observationdatafile):

    #Define global variables that are used in other functions
    global hydrograph, simulation_timestep, sim_time, P_prime
```

```
    P_prime = []

    #For each guess
    for trialfile in filelist:

        #Initialize the lists that will be populated by PySWMM
        hydrograph = []
        sim_time = []

        #Open each trialfile with PySWMM
        with Simulation(trialfile) as sim:

            #Initialize the nodes that will be perused to find the root
            node_object = Nodes(sim)

            #Set the root location within PySWMM
            root_location = node_object[root]

            #Specify that PySWMM should report simulation values at the same time
resolution as the observed data
            simulation_timestep = time_difference.total_seconds()
            sim.step_advance(simulation_timestep)

            #For each time step in a SWMM simulation with PySWMM
            for step in sim:

                #Append the current time
                sim_time.append(sim.current_time)

                #Append the hydrograph value
                hydrograph.append(root_location.total_inflow)

        #Determine the objective function values for each trial file
        objFunc = [normalizedpeakerror(), normalizedvolumeerror(), nashsutcliffe(),
L2.L2norm(trialfile)]

        #Append the objFunc list to a transformed list of guesses P_prime
        P_prime.append(objFunc)
    return

#Compute the aggregate function of objective functions
def aggregateFunction():

    #Initialize a global list aggFunc to be sorted in another function
    global aggFunc
    aggFunc = []

    #For each guess in a generation of P', compute the aggregate function
```

```
    for objFunc in P_prime:
        aggFunc.append(objFunc[0]*weights[0] + objFunc[1]*weights[1] +
objFunc[2]*weights[2] + objFunc[3]*weights[3])

    #Return the list of aggregate function values
    return(aggFunc)

#Create a list ranking the aggregate functions values for a given generation
def rankP_prime():

    #Define a dummy variable x that is equal to the list of aggregate function
values
    x = aggregateFunction()

    #Sort x from low to high
    seq = sorted(x)

    #Create a new list, index, that is the rank of the aggregate function value for
a given guess relative to the other
    # guesses within that generation
    index = [seq.index(v) for v in x]

    #Return this indexed list
    return(index)
```

C.5 Generations.py

Generations.py contains the workflow for the abbreviated NSGA-II algorithm explained by §2.4. It assesses the ranked guesses, culls the poor performers, and repopulates a new generation with mutations of the best performing guesses. Generations.py references both Objective_functions.py to access the ranked list of guesses as well as CreateGuesses.py to produce each subsequent generation of guesses.

```
#Import Objective_functions to access the rankP_prime() function
import Objective_functions

#Import CreateGuesses to access read_initial_parameters() and
compile_initial_guess() functions
import CreateGuesses

#Initialize lists of input files that comprise the current and next generation
filelist = ['trialfile01.inp', 'trialfile02.inp', 'trialfile03.inp',
'trialfile04.inp', 'trialfile05.inp',
            'trialfile06.inp', 'trialfile07.inp',
            'trialfile08.inp', 'trialfile09.inp', 'trialfile10.inp',
'trialfile11.inp', 'trialfile12.inp',
            'trialfile13.inp', 'trialfile14.inp',
            'trialfile15.inp', 'trialfile16.inp', 'trialfile17.inp',
'trialfile18.inp', 'trialfile19.inp',
            'trialfile20.inp', 'trialfile21.inp',
            'trialfile22.inp', 'trialfile23.inp', 'trialfile24.inp',
'trialfile25.inp', 'trialfile26.inp',
            'trialfile27.inp', 'trialfile28.inp',
            'trialfile29.inp', 'trialfile30.inp', 'trialfile31.inp',
'trialfile32.inp', 'trialfile33.inp',
            'trialfile34.inp', 'trialfile35.inp',
            'trialfile36.inp', 'trialfile37.inp', 'trialfile38.inp',
'trialfile39.inp', 'trialfile40.inp',
            'trialfile41.inp', 'trialfile42.inp',
            'trialfile43.inp', 'trialfile44.inp', 'trialfile45.inp',
'trialfile46.inp', 'trialfile47.inp',
            'trialfile48.inp', 'trialfile49.inp',
            'trialfile50.inp', 'trialfile51.inp', 'trialfile52.inp',
'trialfile53.inp', 'trialfile54.inp',
            'trialfile55.inp', 'trialfile56.inp',
            'trialfile57.inp', 'trialfile58.inp', 'trialfile59.inp',
'trialfile60.inp', 'trialfile61.inp',
            'trialfile62.inp', 'trialfile63.inp',
            'trialfile64.inp', 'trialfile65.inp', 'trialfile66.inp',
'trialfile67.inp', 'trialfile68.inp',
            'trialfile69.inp', 'trialfile70.inp',
            'trialfile71.inp', 'trialfile72.inp', 'trialfile73.inp',
'trialfile74.inp', 'trialfile75.inp',
            'trialfile76.inp', 'trialfile77.inp',
            'trialfile78.inp', 'trialfile79.inp', 'trialfile80.inp',
'trialfile81.inp', 'trialfile82.inp',
            'trialfile83.inp', 'trialfile84.inp',
            'trialfile85.inp', 'trialfile86.inp', 'trialfile87.inp',
'trialfile88.inp', 'trialfile89.inp',
            'trialfile90.inp', 'trialfile91.inp',
            'trialfile92.inp', 'trialfile93.inp', 'trialfile94.inp',
```

```
'trialfile95.inp', 'trialfile96.inp',
            'trialfile97.inp', 'trialfile98.inp',
            'trialfile99.inp', 'trialfile100.inp']
Qfilelist = ['trialfile101.inp', 'trialfile102.inp', 'trialfile103.inp',
'trialfile104.inp', 'trialfile105.inp',
            'trialfile106.inp', 'trialfile107.inp',
            'trialfile108.inp', 'trialfile109.inp', 'trialfile110.inp',
'trialfile111.inp', 'trialfile112.inp',
            'trialfile113.inp', 'trialfile114.inp',
            'trialfile115.inp', 'trialfile116.inp', 'trialfile117.inp',
'trialfile118.inp', 'trialfile119.inp',
            'trialfile120.inp', 'trialfile121.inp',
            'trialfile122.inp', 'trialfile123.inp', 'trialfile124.inp',
'trialfile125.inp', 'trialfile126.inp',
            'trialfile127.inp', 'trialfile128.inp',
            'trialfile129.inp', 'trialfile130.inp', 'trialfile131.inp',
'trialfile132.inp', 'trialfile133.inp',
            'trialfile134.inp', 'trialfile135.inp',
            'trialfile136.inp', 'trialfile137.inp', 'trialfile138.inp',
'trialfile139.inp', 'trialfile140.inp',
            'trialfile141.inp', 'trialfile142.inp',
            'trialfile143.inp', 'trialfile144.inp', 'trialfile145.inp',
'trialfile146.inp', 'trialfile147.inp',
            'trialfile148.inp', 'trialfile149.inp',
            'trialfile150.inp', 'trialfile151.inp', 'trialfile152.inp',
'trialfile153.inp', 'trialfile154.inp',
            'trialfile155.inp', 'trialfile156.inp',
            'trialfile157.inp', 'trialfile158.inp', 'trialfile159.inp',
'trialfile160.inp', 'trialfile161.inp',
            'trialfile162.inp', 'trialfile163.inp',
            'trialfile164.inp', 'trialfile165.inp', 'trialfile166.inp',
'trialfile167.inp', 'trialfile168.inp',
            'trialfile169.inp', 'trialfile170.inp',
            'trialfile171.inp', 'trialfile172.inp', 'trialfile173.inp',
'trialfile174.inp', 'trialfile175.inp',
            'trialfile176.inp', 'trialfile177.inp',
            'trialfile178.inp', 'trialfile179.inp', 'trialfile180.inp',
'trialfile181.inp', 'trialfile182.inp',
            'trialfile183.inp', 'trialfile184.inp',
            'trialfile185.inp', 'trialfile186.inp', 'trialfile187.inp',
'trialfile188.inp', 'trialfile189.inp',
            'trialfile190.inp', 'trialfile191.inp',
            'trialfile192.inp', 'trialfile193.inp', 'trialfile194.inp',
'trialfile195.inp', 'trialfile196.inp',
            'trialfile197.inp', 'trialfile198.inp',
            'trialfile199.inp', 'trialfile200.inp']
Unionsetlist = filelist + Qfilelist
```

```
#Specify the observation data set, the root location, and the objective function
weights
observationdatafile = "trial_observation.dat"
root = "18"
weights = [0.25, 0.25, 0.25, 0.25]

#To create the 2nd generation, for each 1st generation file
for infile in filelist:

    #Read the parameters
    CreateGuesses.read_initial_parameters(infile)

    #And compile the initial guess
    CreateGuesses.compile_initial_guess()

    #Specify the corresponding 2nd generation file, and "make a generation" on that
single file that is only 1 file long
    # compiling 100 "make a generation"s gives you a full 2nd generation.
    Qfile = [Qfilelist[filelist.index(infile)]]
    CreateGuesses.create_generation(infile, Qfile)

#Write the Objective Function values for the #1-ranked guess for each generation
with open('Ob_Func.txt', 'w') as file:

    #Write the Headers
    file.write("NPE        NVE        NSEm        NED \n")

    #For 100 generations
    for iteration in range(100):
        #Determine the objective function values for each file in the 2 generations
being compared
        Objective_functions.objectivefunctions(Unionsetlist, observationdatafile)

        #Initialize the nextgenlist and the survivinglist, which will be populated
with files that are to be replaced
        # or to be maintained, respectively.
        nextgenlist = []
        survivinglist = []

        #For each guess in the ranked list of guesses,
        for guess in Objective_functions.rankP_prime():

            #If the guess is ranked worse than 100th, add the corresponding input
file to the nextgenlist
            if guess >= 100:

nextgenlist.append(Unionsetlist[Objective_functions.rankP_prime().index(guess)])
```

```
            #If the guess is ranked 100th or better, add the corresponding input
file to the survivinglist
            else:

survivinglist.append(Unionsetlist[Objective_functions.rankP_prime().index(guess)])

            #If the guess is ranked 1st, print the objective functions to the file,
and print the name of the file
            # and aggregate function value to the screen for ease of tracking
progress
            if guess == 0:

print(Objective_functions.aggFunc[Objective_functions.rankP_prime().index(guess)])
                print(Unionsetlist[Objective_functions.rankP_prime().index(guess)])
                file.write("{:.04f}     {:.04f}     {:.04f}
{:.04f}\n".format(Objective_functions.P_prime[

Objective_functions.rankP_prime().

index(guess)][0], Objective_functions.

P_prime[Objective_functions.

rankP_prime().index(guess)][1],

Objective_functions.P_prime[

Objective_functions.rankP_prime().

index(guess)][2], Objective_functions.

P_prime[Objective_functions.

rankP_prime().index(guess)][3]))

        #For each file that is being kept to the next generation
        for goodfile in survivinglist:
            #Reread the new inital parameters and compile the initial guess
            CreateGuesses.read_initial_parameters(goodfile)
            CreateGuesses.compile_initial_guess()

            #The nextgenlist corresponding to the goodfile is mutated to create the
child for the next generation
            Qfile = [nextgenlist[survivinglist.index(goodfile)]]
            CreateGuesses.create_generation(goodfile, Qfile)
```

D Formulation of SWMMCALPY2.0's Search and Decision Criteria

The methodology presented in Chapter 2 can be said to describe the initial development of SWMMCALPY, or SWMMCALPY1.0. Several decisions were made to reduce the complexity of the algorithm and expedite the implementation of the code behind it. Presented in this Appendix is the mathematical blueprint for the implementation of SWMMCALPY2.0, or the next iteration of SWMMCALPY that includes the full search, sort, and decision making criteria for executing the NSGA-II algorithm. For each decision, the SWMMCALPY1.0 approximation is restated, and the plans for SWMMCALPY2.0 are summarized, with the logical equations presented.

D.1 Non-dominated Sorting

By and large, SWMMCALPY2.0 is identical to SWMMCALPY1.0. Divergence first occurs when the first generation of $P \bigcup Q$ is sorted for non-dominance. In SWMMCALPY1.0, this is done using a single, aggregated objective function shown in Eq. 2.9. Using this method, non-dominance is essentially approximated by a weighted distance from the origin in the feasible performance space. A commonly used approach (Barco et al., 2008), this aggregate function does a good job differentiating between points that are clearly dominated and those that are near the Pareto front, but has difficulty identifying the correct solution set for problems in which the Pareto front has a general shape (i.e. not convex) (Deb et al., 2002).

In SWMMCALPY2.0, however, the full non-dominance criteria are applied to each generation posed by the NSGA-II algorithm. Spelled out in Deb et al. (2002), this algorithm establishes a Pareto-nearness ranking criteria by which the

guesses are evaluated. The surviving set, new P, is filled with the highest ranking guesses. Ties are broken using a distance clustering technique that is also presented. The logic behind the "fast non-dominated sort" algorithm is shown in Figure D.2 and subsequently paraphrased.

```
fast-non-dominated-sort(P)
for each p ∈ P
   S_p = ∅
   n_p = 0
   for each q ∈ P
      if (p ≺ q) then                    If p dominates q
         S_p = S_p ∪ {q}                 Add q to the set of solutions dominated by p
      else if (q ≺ p) then
         n_p = n_p + 1                   Increment the domination counter of p
   if n_p = 0 then                       p belongs to the first front
      p_rank = 1
      F_1 = F_1 ∪ {p}
i = 1                                    Initialize the front counter
while F_i ≠ ∅
   Q = ∅                                 Used to store the members of the next front
   for each p ∈ F_i
      for each q ∈ S_p
         n_q = n_q − 1
         if n_q = 0 then                 q belongs to the next front
            q_rank = i + 1
            Q = Q ∪ {q}
   i = i + 1
   F_i = Q
```

Figure D.2: "Fast Non-dominated Sort" algorithm from Deb et al. (2002).

The fast non-dominated sort algorithm works by essentially distilling a generation of guesses into classes or "shells" (think electron shells on an atomic model) based on non-dominance. For each generation, every guess is compared to every other guess. If, for a given P_i, there does not exist a P_j such that $P_j < P_i$ for all objective functions, that means that P_i is non-dominated. This criteria is logically

expressed in Eq. D.1

$$
\begin{aligned}
&For\ P_i\ in\ P: \\
&\quad For\ P_j\ in\ P\ where\ j \neq i: \\
&\quad\quad if\ F_1(P_j) < F_1(P_i) \\
&\quad\quad and\ if\ F_2(P_j) < F_2(P_i) \\
&\quad\quad ... \\
&\quad\quad and\ if\ F_n(P_j) < F_n(P_i) \\
&\quad Then\ P_i\ is\ NOT\ non-dominated
\end{aligned}
\tag{D.1}
$$

Eq. D.1 expands on the logic presented in Figure D.2. Should any of the objective function comparison tests fail every P_j for a given P_i that means P_i is non-dominated, and its genetic information should persist to the next generation. These generationally non-dominated guesses comprise the first shell. It should be noted that the objective functions are sought to be minimized, so smaller values of F_i are desirable.

Conversely, should P_i be found to be dominated, it is added to the S_p set indicated in Figure D.2. Once set P has been entirely searched, the search for non-dominance is repeated on set S_p. Non-dominated guesses from set S_p comprise the second shell of guesses, and the remainder are passed to the next set, which will be called $S_{p=1}$. This cascading search for non-dominance continues until the surviving generation has been filled. In the case of SWMMCALPY, where the generation size is 100, this means that 100 guesses from the various non-dominated sets, P first, then S_p, then $S_{p=1,2,...,n}$ are included in the next iteration of NSGA-II genetic

process. Deb et al. Deb et al. (2002) lauds this search algorithm as the "fast non-dominated sort" because it has computational complexity of $O(MN^2)$, where M is the number of sortings, or number of sets, P and S_{p_i}, and N is the number of guesses in a generation.

The question arises from this arbitrary generation size cutoff: what happens if the generation size is reached halfway through a set? Should that occur, a clustering metric is applied to ensure that diversity among solutions is used as a tie-breaker. When each guess is being compared to each other guess on the basis of the objective functions, the distance between the two guesses in the feasible performance space is also evaluated. When a non-dominance approximating shell exceeds the allotted spaces within the next generation, the distance vector for each guess becomes important. In an attempt to promote diversity of solutions, the guesses within the set are further sorted based on the cumulative distance to the nearest two points, from highest to lowest. In doing so, guesses in regions that are less densely populated are more likely to be included in the surviving set of guesses.

Once the surviving set of guesses is filled, the child generation is produced the same way in SWMMCALPY2.0 as in SWMMCALPY1.0, by leveraging the genetic parameters given in Table 2.4.

D.2 Stop Criteria

In §2.4, options for a stop criteria are discussed. For the simple aggregate objective function sorting approach used by SWMMCALPY1.0, a number of iterations passed – rather than a determination of convergence – is sufficient. For the fast non-dominated sort detailed in §D.1, an estimate of convergence is perhaps a more

robust way of determining if the NSGA-II algorithm has approached the Pareto front. Shan and Wang (2005) suggests that for each generation, the previous generation can be consulted for the answer.

The NSGA-II, being an elitist algorithm, ensures that guesses with the best performance persist to the next generations. Shan and Wang (Shan & Wang, 2005) posits that this elitism can be leveraged for determining whether the NSGA-II algorithm has ceased progressing. The stop criteria suggested by Shan and Wang (Shan & Wang, 2005) is simple and the logic is paraphrased as such: if the surviving set for a given generation contains 95% of the same population of guesses as the previous generation, exit. Essentially, if only 5% of the population of the surviving guesses has been turned over from the last generation that is a good indicator that a large portion of the guesses are truly non-dominated, and are near the Pareto front.

D.3 Selecting One Solution Along Pareto Front

In SWMMCALPY1.0, to recap §2.5 the selection of a final solution is actually fairly straightforward. Because the objective function weights behave as coefficients in the aggregate function, the final solution that is returned to the engineer is simply the parameter set, or guess, with the smallest value of the aggregate function after the iterations stopping criteria has been reached.

For SWMMCALPY2.0, the objective function weights do not play a role until the very end. The NSGA-II algorithm and approach to determining the Pareto front do not rely on the weights. The effect of this is that the Pareto front is static for a given problem. Where the engineer's preference becomes important is identifying a single solution from among these Pareto-approximating solutions.

An optimal angle, or more specifically, an optimal vector of angles is defined based on a vector of the objective function weights to each of objective function axis.

$$\Theta = (\theta_1, \theta_2, \theta_3, \theta_4) \tag{D.2}$$

$$\theta_i = arccos\Big(\frac{W \cdot e_i}{|W||e_i|}\Big) \tag{D.3}$$

$$W = (w_1, w_2, w_3, w_4) \tag{D.4}$$

Eq. D.2 contains the optimal angle vector Θ, which will be used to distinguish between solutions in on the Pareto front. The equation by which each component of the optimal angle vector is calculated is presented in Eq. D.3 [CITATION]. W corresponds to the vector of the objective function weights, shown in Eq. D.5, while e_i indicates the unit vector along each of the objective function axis. This formulation is given specifically for 4 objective functions, but can be very easily generalized to more or fewer objective functions.

Using Eq. D.3, the angle vector corresponding to each of the parameter sets in the final generation from the NSGA-II portion of SWMMCALPY can be calculated. Vector W would simply need to be supplanted with a new vector, $F^{'}$. The components of $F^{'}$ are the values of the objective functions, or, more simply, the coordinate in the feasible performance space for the guess under scrutiny. $F^{'}$ is similar to the aggregate function F, but in vector form and sans weight coefficients.

$$F^{'} = (f_1, f_2, f_3, f_4) \tag{D.5}$$

The transformation from coordinates in the feasible performance space into feasible "angle" space allows the Euclidean distance between each solution and the coordinate of the optimal angle vector to be calculated. The Pareto front solution whose angle is nearest to the optimal angle is returned to the engineer as the calibrated parameter set.

These three alterations to the NSGA-II execution within the SWMMCALPY2.0 workflow constitute a significant upgrade in sophistication, as well as an increased flexibility in handling non-convex Pareto fronts. The effect of these alterations on the final solution is discussed in §5.2.

E Detailed Report on Brentwood Test Case Preprocessing by Brittany Hornik

E.1 Objectives of File Preparation

To validate that the SWMMCALPY program functions as intended, data files from an area with known rainfall had to be prepared and tested. The data chosen was from the neighborhood of Brentwood from 2012-2014 because of our ability to access the data files through Dr. Brandon Klenzendorf at the consulting company Geosyntec, Inc. The Brentwood files were prepared to accomplish two key objectives: evaluate the "uncalibrated" input files based on four objective functions used by SWMMCALPY and uncalibrated the input files to use as a starting point for the SWMMCALPY routine.

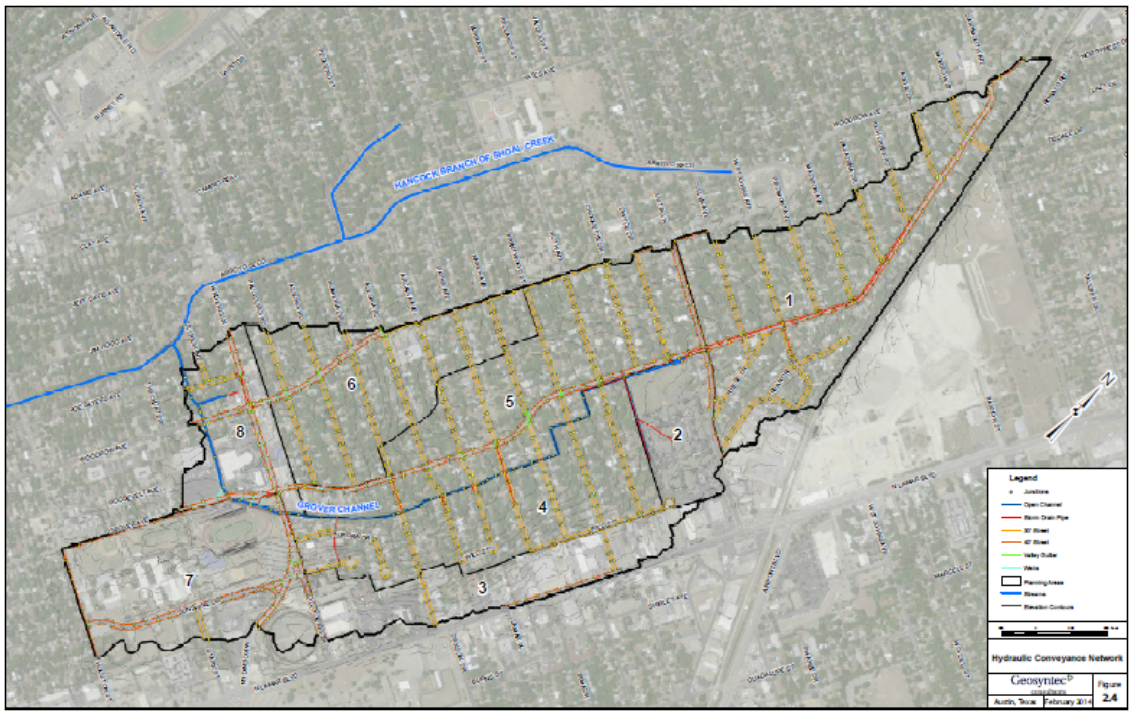


Figure E.3: Key Elements of the Brentwood Drainage System. From Geosyntec (2017)

E.2 Background on Brentwood

The data that was used to evaluate the SWMMCALPY routine was from Brentwood, a neighborhood in central Austin, Texas. A consulting firm, Geosyntec, was hired by the City of Austin to determine solutions to flooding, erosion, and water quality problems in the neighborhood. To properly accomplish this, they developed a model in PCSWMM of the drainage system in Brentwood involving elements such as subcatchments, junctions, inlets, conduits, and rain gages. Figure E.3 depicts the key elements of the model superimposed on a physical map of the area being investigated.

To create an accurate model, information about the topography and drainage

features had to be input into PCSWMM. Much of the data was collected from a GIS database, Shapefile, or other databases. Information that could not be found elsewhere was gathered in the field and input into the PCSWMM model. The forcing data inputs for precipitation calibration came directly from Brentwood rain gages, BW1 and BW2. From Figure E.3, BW1 acted as both a rain gage and the outfall for the entire Brentwood system.

The Brentwood model was developed in PCSWMM and calibrated using Sensitivity-based Radio Tuning Calibration (SRTC). This tool functions to adjust the calibrated parameters with the goal of reducing the variability between the observed and simulated rainfall events over the period of interest, particularly the difference in peak flow rates and total runoff volume for the Brentwood model. The SWMMCALPY calibration method will be analyzed in comparison to the SRTC calibrated parameters to verify the routine against a valid case study by professional engineers.

E.3 Evaluation of Calibrated Brentwood Input Files

For the evaluation of the calibrated Brentwood files, a simulation of the storm with the greatest volume of rainfall was run in SWMM 5. Before running the simulation, the date with the largest rainfall had to be determined. To accomplish this, python code was developed that read the BW1_raingage data file and sorted the dates from least to greatest rainfall as determined by a tipping bucket. It was determined that October 13, 2013 was the day that experienced the heaviest rainfall. With the largest volume storm found, the spin-up process could begin.

Spin-up involved clipping the forcing files to determine the time required prior to the event of interest that no significant changes were made to the resulting

hydrographs. The largest rain storm was from 12AM to 2PM on October 13, 2013. This time frame was used as the reference. Copies of the BW1_raingage and BW2_raingage text files were made so that the data in them could be altered. First, the files were clipped so that they only contained data from 12AM to 2PM on October 13, 2013. The Brentwood model with these newly altered files were then used to run a simulation on SWMM 5 between the aforementioned times and dates. Since BW1 acted as the outlet for the neighborhood, the computed flow rates through the outlet and their corresponding times since the start of the simulation were downloaded and input into an Excel spreadsheet. Using 0 as midnight on October 13, 2013, the hydrograph for the time period was plotted. The rain gage files were clipped to four different times to contain rainfall data from 1.5, 4, 6, 8, 12, 24, and 48 hours prior to 12AM on October 13, 2013. The outlet flow rates and times were exported to Excel. To normalize the data that started prior to midnight, the 0-time values needed to align. This was accomplished by dividing the hours prior to midnight by 24, subtracting that value from the days, and adding the hours. Once normalized the hydrographs produced the plot in Figure E.4.

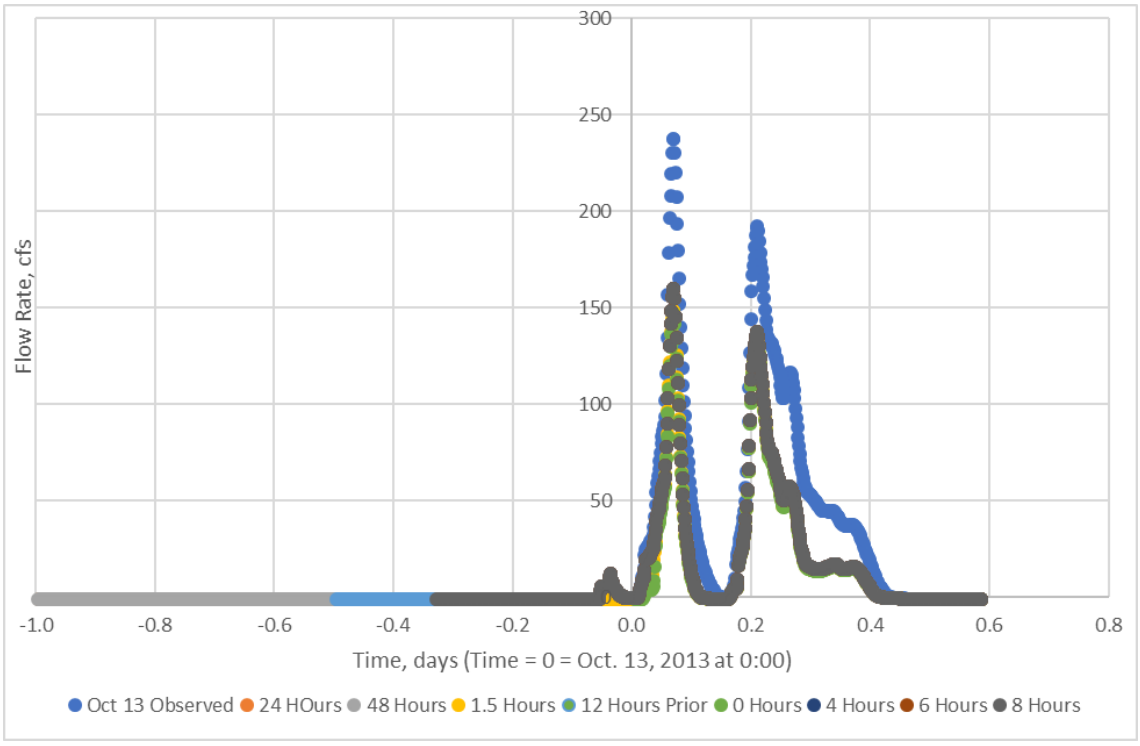


Figure E.4: Hydrographs of Rainfall Data Clipped 48, 24, 12, 8, 6, 4, 1.5, and 0 Hours Prior to 12AM 10/13/13.

While each hydrograph generally follows the same path, some variations exist between graphs. For example, the peak flow rates were inconsistent from hydrograph to hydrograph.

A modified Nash-Sutcliffe efficiency index was applied to the data to determine the spin-up time required for the modeled hydrograph to estimate the observed hydrograph. The data were compared in sequential order beginning with the comparison of data from 10/13/13 from 12AM-2PM and data from 10/12/13 from 10:30PM to 10/13/13 at 2PM. Eq. 2.7 is the modified Nash-Sutcliffe efficiency index.

The NSE was computed for each test. Flow rates were only compared along the same time interval. For example, when comparing the October 13 reference

with the flow rates produced from beginning the simulation an hour and a half prior to midnight, the flow rates used were those that shared the same time frame from midnight to 2PM. With the NSE calculated for each test prior to the reference time, the derivatives between the time differences were found and plotted in Figure E.5. This was accomplished by taking the difference between the sequential times prior and dividing by the time difference. Table E.1 lists the computed NSE and the derivative between the tests.

Table E.1: Modified Nash-Sutcliffe Efficiencies of Tested Spin-Up Times.

Time Prior to Reference (hrs)	NSE	∂NSE/∂t
0	0.660	-
1.5	0.670	0.006
4	0.717	0.019
6	0.717	0
8	0.717	0
12	0.717	0
24	0.717	0
48	0.717	0

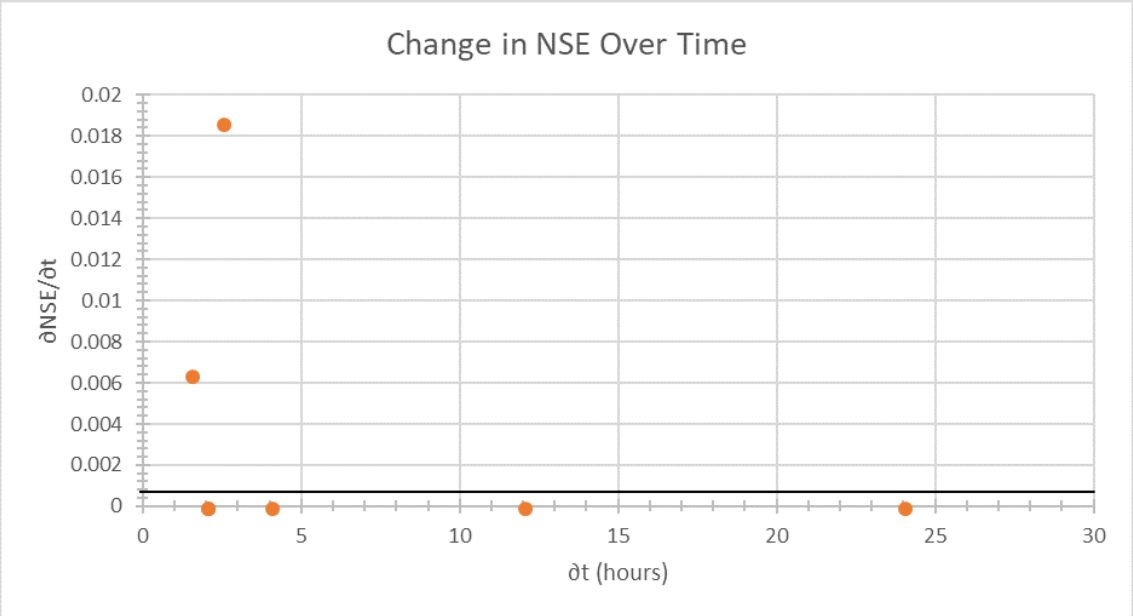


Figure E.5: Sensitivity Parameter vs Time Increment for simulations with rainfall data clipped to several hours prior to 12AM on October 13, 2013.

It can be surmised from Figure E.5 that the derivative of the modified NSE with respect to time is relatively constant after a time difference of 4 hours. With the spin-up time determined, the Brentwood files could now be properly prepared for SWMMCALPY.

E.4 Preparation of Uncalibrated Brentwood Input Files

A crucial step in the calibration process is the verification of the calibrated results with observed data. Therefore, the objective functions produced from SWMMCALPY must be similar to those produced from the comparison of the uncalibrated and calibrated Brentwood input files. The Brentwood model determined the calibrated parameters from information found in the field or in databases. These parameters were calibrated based on the information from Table E.2 in the

technical report provided by Geosyntec Consultants.

Table E.2: Summary of Subcatchment Parameters Analyzed for Calibration obtained from Geosyntec (2017).

Parameter	Initial Value	Range	Calibrated Value
Subcatchment Width (ft)	Varies	Varies	-62% for non-ROW[1] -44% for ROW
Impervious Cover (%)	Varies	Varies	-11% for non-ROW[2] -5% for ROW[2]
N for Impervious Cover	0.016	0.008 to 0.032	0.029
N for Pervious Cover	0.2	0.1 to 0.4	0.35
Depression Storage for Impervious Cover (in.)	0.05	0.025 to 0.1	0.09
Depression Storage for Pervious Cover (in.)	0.07	0.035 to 0.14	0.123
Percent Impervious Cover with Zero Depression (%)	21.7	12.7 to 36.8	23.9
Percent Routed (%)	40 for non-ROW 0 for ROW	20 to 80	28.6 for non-ROW 0 for ROW
Suction Head (in.)	8	4 to 16	10.8
Saturated Hydraulic Conductivity (in/hr)	0.4	0.133 to 1.20	0.27
Initial Moisture Deficit	0.2	0.1 to 0.4	0.37

To obtain the objectives functions, these parameters had to be uncalibrated. The input file "001_Brentwood_1D_Existing_Monitoring_rev1" was converted to a .csv file and imported into Microsoft Excel to efficiently alter the desired parameters. The parameters of interest for the subcatchments were the "Subcatchment Width" and "Impervious Cover", and the parameters "N for Impervious Cover", "N for Pervious Cover", "Depression Storage for Impervious Cover", "Depression Storage for Pervious Cover", and "Percent Impervious Cover with Zero Depression" were altered for the subareas. While the subarea parameters were reverted to their initial values in Table E.2, the subcatchment parameters had to recalculated based on their calibrated values. The "Subcatchment Width" was uncalibrated

using the following formula.

$$Initial\ Width = 2 * Calibrated\ Width \tag{E.1}$$

The "Impervious Cover" was uncalibrated in a similar manner.

$$Initial\ Impervious\ Cover = 1.1 * Calibrated\ Impervious\ Cover \tag{E.2}$$

The above arithmetic was applied to the specified parameters for each of the 619 subcatchments.

Reverting the calibrated parameters back to their initial values created a SWMM input file of the Brentwood model. This uncalibrated file could be compared to the calibrated file by calculating the L2-norm. Furthermore, SWMMCALPY could perform the recalibration of this input file and return its calculated L2-norm. These computed objective functions, once compared, could determine the effectiveness and validity of SWMMCALPY.

E.5 Verification

The verification process is intended to determine whether SWMMCALPY can achieve its purpose as a method for calibration. It involves the calculation of the objectives functions based on the observed data and the computed SWMM data at the outfall of the Brentwood case.

The previously determined spin-up time was used to clip the observational data to the desired October 13, 2013 event. This outfall data was then compared

to the simulated outfall data for the same time frame of the event to compute the four objective functions: peak error, volume error, Nash-Sutcliffe Efficiency, and L2-norm.

Microsoft Excel was utilized to transparently calculate the objective functions. The formulas for the peak error, volume error, Nash-Sutcliffe Efficiency, and L2-norm are presented below. The peak error in Eq. 2.1 compared the maximum total inflows of the observer and simulated data.

Volume error required the integration of the hydrographs produced for the observed and simulated data. To simplify, the trapezoid method was employed to incrementally find the areas between each time interval of one minute. The NVE calculation is given by Eq. 2.2 and Eq. 2.3.

The Nash Sutcliffe Efficiency objective function was determined as it was previous when calculating the required spin-up time, Eq. 2.7. As for the L2-norm, the "distance" in the n-dimensional space between the calibrated and uncalibrated input files had to be normalized by the "size" of the calibrated and uncalibrated parameters. This is shown in Eq. 2.8.

The evaluation of the Brentwood model was accomplished in accordance with the objective functions computed by SWMMCALPY. The final computed objectives functions for the October 13, 2013 are presented in Table 3.3. The validity of SWMMCALPY is to be determined from the comparison of the objectives functions produced by SWMMCALPY from the uncalibrated Brentwood input file and the objectives functions computed from the above calculations.

E.6 Conclusion

The validation of the SWMMCALPY routine required that its calibrated parameters be compared to a calibration method that is already verified and validated. In this case, the SRTC in PCSWMM was the professional calibration tool against which the SWMMCALPY parameters were compared. Geosyntec provided rainfall data from 2012-2014 as well as a SWMM model for the Austin neighborhood of Brentwood. The contributed files had to undergo pre-processing to prepare for SWMMCALPY.

During pre-processing the rainfall event was selected, the spin-up time for the single storm was determined, the input file was uncalibrated, and the objective functions of the SWMM simulation of the Brentwood data were calculated. A python script was written to sort the three years' worth of rainfall from days with the greatest rainfall to days with the least. October 13, 2013 was the day of highest rainfall and the event of interest for the rest of pre-processing. To find the spin-time, simulations were run at various times prior to midnight of the October 13 storm and the flow rates were compared using a modified Nash-Sutcliffe efficiency index. The spin-up time was determined to be four hours. With this information, the simulation run for the October 13 storm was compared to the clipped October 13 storm from a continuous simulation to find the peak error, volume error, and modified Nash-Sutcliffe efficiency. The Brentwood input file was uncalibrated back to its original values for several parameters. This input file was then compared to the calibrated input file to calculate the L2-norm. The four objectives functions computed are ready to be measured against objective functions produced by SWMMCALPY on the uncalibrated input file.

References

Alamdari, N., Sample, D., Steinberg, P., Ross, A., & Easton, Z. (2017, June). Assessing the Effects of Climate Change on Water Quantity and Quality in an Urban Watershed Using a Calibrated Stormwater Model. *Water*, *9*(12), 464. Retrieved 2018-03-07, from `http://www.mdpi.com/2073-4441/9/7/464` doi: 10.3390/w9070464

Angelakis, A. N. (2017, October). Urban waste- and stormwater management in Greece: past, present and future. *Water Science and Technology-Water Supply*, *17*(5), 1386–1399. (WOS:000417944300021) doi: 10.2166/ws.2017.042

Audusse, E., Bouchut, F., Bristeau, M.-O., Klein, R., & Perthame, B. (2004, January). A Fast and Stable Well-Balanced Scheme with Hydrostatic Reconstruction for Shallow Water Flows. *SIAM Journal on Scientific Computing*, *25*(6), 2050–2065. Retrieved 2018-04-03, from `http://epubs.siam.org/doi/10.1137/S1064827503431090` doi: 10.1137/S1064827503431090

Baiamonte, G., & Singh, V. P. (2017, July). Modeling the probability distribution of peak discharge for infiltrating hillslopes. *Water Resources Research*, *53*(7), 6018–6032. (WOS:000407895000046) doi: 10.1002/2016WR020109

Bao, H., Wang, L., Zhang, K., & Li, Z. (2017, July). Application of a de-

veloped distributed hydrological model based on the mixed runoff generation model and 2d kinematic wave flow routing model for better flood forecasting: The developed GMKHM-2d model for better flood forecasting. *Atmospheric Science Letters*, *18*(7), 284–293. Retrieved 2018-04-04, from http://doi.wiley.com/10.1002/asl.754 doi: 10.1002/asl.754

Barco, J., Wong, K. M., & Stenstrom, M. K. (2008). Automatic calibration of the US EPA SWMM model for a large urban catchment. *Journal of Hydraulic Engineering*, *134*(4), 466–474.

Bastidas, L. A., Gupta, H. V., Sorooshian, S., Shuttleworth, W. J., & Yang, Z. L. (1999). Sensitivity analysis of a land surface scheme using multicriteria methods. *Journal of Geophysical Research: Atmospheres*, *104*(D16), 19481–19490.

Beven, K., & Binley, A. (1992, July). The future of distributed models: Model calibration and uncertainty prediction. *Hydrological Processes*, *6*(3), 279–298. Retrieved 2018-04-06, from http://doi.wiley.com/10.1002/hyp.3360060305 doi: 10.1002/hyp.3360060305

Box, M. J. (1965, April). A New Method of Constrained Optimization and a Comparison With Other Methods. *The Computer Journal*, *8*(1), 42–52. Retrieved 2018-04-06, from https://academic.oup.com/comjnl/article-lookup/doi/10.1093/comjnl/8.1.42 doi: 10.1093/comjnl/8.1.42

Campolongo, F., & Braddock, R. (1999, January). Sensitivity analysis of the IMAGE Greenhouse model. *Environmental Modelling & Software*, *14*(4), 275–282. Retrieved 2018-04-06, from http://linkinghub.elsevier.com/retrieve/pii/S1364815298000796 doi: 10.1016/S1364-8152(98)00079-6

CHI Water. (2018). *Introduction to PCSWMM Workshop*. CHI Water.

Chow, V. T. (1959). *Open-Channel Hydraulics*. McGraw-Hill Book Company.

Cipolla, S. S., Maglionico, M., & Stojkov, I. (2016, October). A long-term hydrological modelling of an extensive green roof by means of SWMM. *Ecological Engineering*, *95*, 876–887. Retrieved 2018-03-07, from http://linkinghub.elsevier.com/retrieve/pii/S0925857416304608 doi: 10.1016/j.ecoleng.2016.07.009

Contributors, V. (2018). *SWMM-EPANET_user_interface.* Retrieved 2018-04-04, from https://github.com/USEPA/SWMM-EPANET_User_Interface (original-date: 2015-12-18T15:41:52Z)

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation*, *6*(2), 182–197.

De Paola, F., Giugni, M., & Pugliese, F. (2016, October). A harmony-based calibration tool for urban drainage systems. *Proceedings of the Institution of Civil Engineers - Water Management*, *171*(1), 30–41. Retrieved 2018-03-07, from https://www.icevirtuallibrary.com/doi/10.1680/jwama.16.00057 doi: 10.1680/jwama.16.00057

Downer, C. W., & Ogden, F. L. (2004, May). GSSHA: Model To Simulate Diverse Stream Flow Producing Processes. *Journal of Hydrologic Engineering*, *9*(3), 161–174. Retrieved 2018-04-03, from http://ascelibrary.org/doi/10.1061/%28ASCE%291084-0699%282004%299%3A3%28161%29 doi: 10.1061/(ASCE)1084-0699(2004)9:3(161)

Engineers, U. A. C. (2016). *HEC-HMS User's Manual, Version 4.2.*

EPA. (2014). *Storm Water Management Model (SWMM)* [Data and Tools]. Retrieved 2018-04-04, from https://www.epa.gov/water-research/storm

-water-management-model-swmm

Feng, Y., & Burian, S. (2016). Improving evapotranspiration mechanisms in the US environmental protection agency's Storm Water Management Model. *Journal of Hydrologic Engineering*, *21*(10), 06016007.

Geosyntec. (2017, January). *Impact of Decentralized Green Stormwater Controls* (Modeling Results Summary). City of Austin Watershed Protection Department.

Granata, F., Gargano, R., & de Marinis, G. (2016, February). Support Vector Regression for Rainfall-Runoff Modeling in Urban Drainage: A Comparison with the EPA's Storm Water Management Model. *Water*, *8*(12), 69. Retrieved 2018-03-07, from http://www.mdpi.com/2073-4441/8/3/69 doi: 10.3390/w8030069

Guan, M., Sillanpää, N., & Koivusalo, H. (2015a, June). Modelling and assessment of hydrological changes in a developing urban catchment: HYDROLOGICAL CHANGES IN A DEVELOPING URBAN CATCHMENT. *Hydrological Processes*, *29*(13), 2880–2894. Retrieved 2018-03-07, from http://doi.wiley.com/10.1002/hyp.10410 doi: 10.1002/hyp.10410

Guan, M., Sillanpää, N., & Koivusalo, H. (2015b). Storm runoff response to rainfall pattern, magnitude and urbanization in a developing urban catchment: Storm Runoff Response to Rainfall Pattern, Magnitude and Urbanization. *Hydrological Processes*, n/a–n/a. Retrieved 2018-03-07, from http://doi.wiley.com/10.1002/hyp.10624 doi: 10.1002/hyp.10624

Herrera, M., Heathcote, I., James, W., & Bradford, A. (2006). Multi-Objective Calibration of SWMM for Improved Simulation of the Hydrologic Regime. *Journal of Water Management Modeling*. Retrieved 2018-03-07, from

https://www.chijournal.org/R225-15 doi: 10.14796/JWMM.R225-15

Huber, W., & Roesner, L. (2012). *The History and Evolution of the EPA SWMM.*

Ibrahim, Y., & Liong, S. (1992, November). Calibration Strategy for Urban Catchment Parameters. *Journal of Hydraulic Engineering*, *118*(11), 1550–1570. Retrieved 2018-04-06, from http://ascelibrary.org/doi/10.1061/%28ASCE%290733-9429%281992%29118%3A11%281550%29 doi: 10.1061/(ASCE)0733-9429(1992)118:11(1550)

Ignatius, A. R., & Jones, J. W. (2018, January). High resolution water body mapping for SWAT evaporative modelling in the Upper Oconee watershed of Georgia, USA. *Hydrological Processes*, *32*(1), 51–65. Retrieved 2018-03-15, from http://onlinelibrary.wiley.com.ezproxy.lib.utexas.edu/doi/10.1002/hyp.11398/abstract doi: 10.1002/hyp.11398

Jetbrains. (2018). *PyCharm: Python IDE for Professional Developers by JetBrains.* Retrieved 2018-04-04, from https://www.jetbrains.com/pycharm/

Johnson, M. T., & Stinchcombe, J. R. (2007). An emerging synthesis between community ecology and evolutionary biology. *Trends in ecology & evolution*, *22*(5), 250–257.

Khu, S. T., & Madsen, H. (2005, March). Multiobjective calibration with Pareto preference ordering: An application to rainfall-runoff model calibration: CALIBRATION WITH PARETO PREFERENCE ORDERING. *Water Resources Research*, *41*(3). Retrieved 2018-03-07, from http://doi.wiley.com/10.1029/2004WR003041 doi: 10.1029/2004WR003041

Knighton, J., Lennon, E., Bastidas, L., & White, E. (2016). Stormwater detention system parameter sensitivity and uncertainty analysis using SWMM. *Journal*

of Hydrologic Engineering, *21*(8), 05016014.

Knighton, J., White, E., Lennon, E., & Rajan, R. (2014, September). Development of probability distributions for urban hydrologic model parameters and a Monte Carlo analysis of model sensitivity: MONTE CARLO ANALYSIS OF URBAN HYDROLOGIC PARAMETERS. *Hydrological Processes*, *28*(19), 5131–5139. Retrieved 2018-03-07, from http://doi.wiley.com/10.1002/hyp.10009 doi: 10.1002/hyp.10009

Krebs, G., Kokkonen, T., Valtanen, M., Koivusalo, H., & Setälä, H. (2013, December). A high resolution application of a stormwater management model (SWMM) using genetic parameter optimization. *Urban Water Journal*, *10*(6), 394–410. Retrieved 2018-03-07, from http://www.tandfonline.com/doi/abs/10.1080/1573062X.2012.739631 doi: 10.1080/1573062X.2012.739631

Krebs, G., Kokkonen, T., Valtanen, M., Setälä, H., & Koivusalo, H. (2014, May). Spatial resolution considerations for urban hydrological modelling. *Journal of Hydrology*, *512*, 482–497. Retrieved 2018-03-07, from http://linkinghub.elsevier.com/retrieve/pii/S0022169414001875 doi: 10.1016/j.jhydrol.2014.03.013

Krebs, G., Kuoppamäki, K., Kokkonen, T., & Koivusalo, H. (2016, January). Simulation of green roof test bed runoff: Simulation of Green Roof Test Bed Runoff. *Hydrological Processes*, *30*(2), 250–262. Retrieved 2018-03-07, from http://doi.wiley.com/10.1002/hyp.10605 doi: 10.1002/hyp.10605

Lappala, E. G., Healy, R. W., & Weeks, E. P. (1987). *DOCUMENTATION OF COMPUTER PROGRAM VS2d TO SOLVE THE EQUATIONS OF FLUID FLOW IN VARIABLY SATURATED POROUS MEDIA* (Water-Resources

Investigations No. 83-4099). U.S. Geological Survey.

Li, C., Liu, M., Hu, Y., Gong, J., & Xu, Y. (2016). Modeling the Quality and Quantity of Runoff in a Highly Urbanized Catchment Using Storm Water Management Model. *Polish Journal of Environmental Studies*, *25*(4), 1573–1581. Retrieved 2018-03-07, from http://www.pjoes.com/doi/10.15244/pjoes/60721 doi: 10.15244/pjoes/60721

Li, J., Li, Y., & Li, Y. (2016, January). SWMM-based evaluation of the effect of rain gardens on urbanized areas. *Environmental Earth Sciences*, *75*(1). Retrieved 2018-03-07, from http://link.springer.com/10.1007/s12665-015-4807-7 doi: 10.1007/s12665-015-4807-7

Liang, Q., & Marche, F. (2009, June). Numerical resolution of well-balanced shallow water equations with complex source terms. *Advances in Water Resources*, *32*(6), 873–884. Retrieved 2018-04-03, from http://linkinghub.elsevier.com/retrieve/pii/S0309170809000396 doi: 10.1016/j.advwatres.2009.02.010

Liong, S., & Ibrahim, Y. (1994, March). Estimation of Peak Flow and Runoff Volume with Response Surface Method. *Journal of Water Resources Planning and Management*, *120*(2), 161–175. Retrieved 2018-04-06, from http://ascelibrary.org/doi/10.1061/%28ASCE%290733-9496%281994%29120%3A2%28161%29 doi: 10.1061/(ASCE)0733-9496(1994)120:2(161)

Liong, S. Y., Chan, W. T., & Lum, L. H. (1991, September). Knowledge-Based System for SWMM Runoff Component Calibration. *Journal of Water Resources Planning and Management*, *117*(5), 507–524. Retrieved 2018-04-06, from http://ascelibrary.org/doi/10.1061/%28ASCE%290733-9496%

281991%29117%3A5%28507%29 doi: 10.1061/(ASCE)0733-9496(1991)117:5(507)

Liong, S.-Y., Chan, W. T., & ShreeRam, J. (1995, August). Peak-Flow Forecasting with Genetic Algorithm and SWMM. *Journal of Hydraulic Engineering*, *121*(8), 613–617. Retrieved 2018-04-06, from http://ascelibrary.org/doi/10.1061/%28ASCE%290733-9429%281995%29121%3A8%28613%29 doi: 10.1061/(ASCE)0733-9429(1995)121:8(613)

Madsen, H. (2000). Automatic calibration of a conceptual rainfall–runoff model using multiple objectives. *Journal of hydrology*, *235*(3-4), 276–288.

Maidment, D. (1993). *Handbook of Hydrology*. McGraw-Hill.

Mantovan, P., & Todini, E. (2006, October). Hydrological forecasting uncertainty assessment: Incoherence of the GLUE methodology. *Journal of Hydrology*, *330*(1-2), 368–381. Retrieved 2018-04-06, from http://linkinghub.elsevier.com/retrieve/pii/S0022169406002162 doi: 10.1016/j.jhydrol.2006.04.046

Mantovan, P., Todini, E., & Martina, M. L. (2007, May). Reply to comment by Keith Beven, Paul Smith and Jim Freer on "Hydrological forecasting uncertainty assessment: Incoherence of the GLUE methodology". *Journal of Hydrology*, *338*(3-4), 319–324. Retrieved 2018-04-06, from http://linkinghub.elsevier.com/retrieve/pii/S0022169407001242 doi: 10.1016/j.jhydrol.2007.02.029

Masseroni, D., & Cislaghi, A. (2016, April). Green roof benefits for reducing flood risk at the catchment scale. *Environmental Earth Sciences*, *75*(7). Retrieved 2018-03-07, from http://link.springer.com/10.1007/s12665-016-5377-z doi: 10.1007/s12665-016-5377-z

McDonnell, B. (2018a, 03). *Personal communication.*

McDonnell, B. (2018b). *PySWMM documentation — pyswmm 0.4.7 documentation.* Retrieved 2018-04-04, from http://pyswmm.readthedocs.io/en/stable/

Mishra, S. K., & Singh, V. (2003). *Soil Conservation Service Curve Number (SCS-CN) Methodology.* Springer Netherlands. Retrieved 2018-04-03, from //www.springer.com/gp/book/9781402011320

Misirlakis, A. (2017, December). *The 7 Most In-Demand Programming Languages of 2018* [Coding Dojo Blog]. Retrieved 2018-04-04, from http://www.codingdojo.com/blog/7-most-in-demand-programming-languages-of-2018/

Moler, C. (2004). *The Origins of MATLAB.* Retrieved 2018-04-04, from https://www.mathworks.com/company/newsletters/articles/the-origins-of-matlab.html

Muleta, M. K., McMillan, J., Amenu, G. G., & Burian, S. J. (2013, October). Bayesian Approach for Uncertainty Analysis of an Urban Storm Water Model and Its Application to a Heavily Urbanized Watershed. *Journal of Hydrologic Engineering*, *18*(10), 1360–1371. Retrieved 2018-03-07, from http://ascelibrary.org/doi/10.1061/%28ASCE%29HE.1943-5584.0000705 doi: 10.1061/(ASCE)HE.1943-5584.0000705

Nash, J. E., & Sutcliffe, J. V. (1970). River flow forecasting through conceptual models part I—A discussion of principles. *Journal of hydrology*, *10*(3), 282–290.

NCIMM. (2018). *Center for Infrastructure Modeling and Management.* Retrieved 2018-03-28, from http://www.ncimm.org/

Neitsch, S., Arnold, J., Kiniry, J., & Williams, J. (2011). *Soil & Water Assessment Tool Theoretical Documentation.* Texas A&M University College of Agriculture and Life Sciences.

Oliveira, T., & Van Noije, W. A. M. (2012, August). Design of Analog Integrated Circuits using Simulated Annealing/Quenching with Crossovers and Particle Swarm Optimization. In M. S. G. Tsuzuki (Ed.), *Simulated Annealing - Advances, Applications and Hybridizations.* In-Tech. Retrieved 2018-03-31, from `http://www.intechopen.com/books/simulated-annealing-advances-applications-and-hybridizations/design-of-analog-integrated-circuits-using-simulated-annealing-quenching-combined-with-population-ba` doi: 10.5772/50384

Python.org. (2018). *Welcome to Python.org.* Retrieved 2018-04-04, from `https://www.python.org/`

Riaño-Briceño, G., Barreiro-Gomez, J., Ramirez-Jaime, A., Quijano, N., & Ocampo-Martinez, C. (2016, September). MatSWMM – An open-source toolbox for designing real-time control of urban drainage systems. *Environmental Modelling & Software*, *83*, 143–154. Retrieved 2018-03-07, from `http://linkinghub.elsevier.com/retrieve/pii/S1364815216301451` doi: 10.1016/j.envsoft.2016.05.009

Rosa, D. J., Clausen, J. C., & Dietz, M. E. (2015). Calibration and Verification of SWMM for low impact development. *Journal of the American Water Resources Association*, *51*(3). doi: 10.1111/jawr.12272

Rossman, L. A. (2016, January). *Storm Water Management Model Reference Manual Volume III – Water Quality.*

Rossman, L. A., & Huber, W. C. (2016, January). *Storm Water Management*

Model Reference Manual Volume I - Hydrology.

Rossman, L. A., & Huber, W. C. (2017, May). *Storm Water Management Model Reference Manual Volume II – Hydraulics.*

Rossman, L. A., James, W., & James, W. R. C. (2010). *Storm Water Managment Model User's Guide.* CHI Press.

Shan, S., & Wang, G. G. (2005). An Efficient Pareto Set Identification Approach for Multiobjective Optimization on Black-Box Functions. *Journal of Mechanical Design*, *127*(5), 866. Retrieved 2018-03-07, from `http://MechanicalDesign.asmedigitalcollection.asme.org/article.aspx?articleid=1448630` doi: 10.1115/1.1904639

Shen, J., & Zhang, Q. (2014). Parameter Estimation Method for SWMM under the Condition of Incomplete Information Based on GIS and RS. *EJGE. Pp*, 6095–6108.

Shinma, T. A., & Reis, L. F. R. (2011). ANALYSIS OF DIFFERENT OBJECTIVE FUNCTIONS SETS APPLIED TO AUTOMATIC CALIBRATION OF THE STORM WATER MANAGEMENT MODEL (SWMM). , 6.

Shinma, T. A., & Reis, L. R. (2014). Incorporating Multi-event and Multi-site Data in the Calibration of SWMM. *Procedia Engineering*, *70*, 75–84. Retrieved 2018-04-03, from `http://linkinghub.elsevier.com/retrieve/pii/S1877705814000125` doi: 10.1016/j.proeng.2014.02.010

Srinivas, R., Singh, A. P., & Deshmukh, A. (2018, January). Development of a HEC-HMS-based watershed modeling system for identification, allocation, and optimization of reservoirs in a river basin. *Environmental Monitoring and Assessment*, *190*(1). Retrieved 2018-03-26, from `http://link.springer.com/10.1007/s10661-017-6418-0` doi: 10.1007/s10661-017-6418-0

Stedinger, J. R., Vogel, R. M., Lee, S. U., & Batchelder, R. (2008, December). Appraisal of the generalized likelihood uncertainty estimation (GLUE) method: APPRAISAL OF THE GLUE METHOD. *Water Resources Research*, *44*(12). Retrieved 2018-04-06, from http://doi.wiley.com/10.1029/2008WR006822 doi: 10.1029/2008WR006822

Sun, N., Hong, B., & Hall, M. (2013, May). Assessment of the SWMM model uncertainties within the generalized likelihood uncertainty estimation (GLUE) framework for a high-resolution urban sewershed: ASSESSMENT OF THE SWMM MODEL UNCERTAINTIES WITHIN THE GLUE FRAMEWORK. *Hydrological Processes*, n/a–n/a. Retrieved 2018-03-07, from http://doi.wiley.com/10.1002/hyp.9869 doi: 10.1002/hyp.9869

Tobio, J. A. S., Maniquiz-Redillas, M. C., & Kim, L. H. (2015, November). Physical design optimization of an urban runoff treatment system using Stormwater Management Model (SWMM). *Water Science and Technology*, *72*(10), 1747–1753. Retrieved 2018-03-07, from http://wst.iwaponline.com/cgi/doi/10.2166/wst.2015.381 doi: 10.2166/wst.2015.381

Versini, P.-A., Ramier, D., Berthier, E., & de Gouvello, B. (2015, May). Assessment of the hydrological impacts of green roof: From building scale to basin scale. *Journal of Hydrology*, *524*, 562–575. Retrieved 2018-03-07, from http://linkinghub.elsevier.com/retrieve/pii/S0022169415001912 doi: 10.1016/j.jhydrol.2015.03.020

Wang, X., Liu, T., & Yang, W. (2012). Development of a robust runoff-prediction model by fusing the Rational Equation and a modified SCS-CN method. *Hydrological Sciences Journal-Journal Des Sciences Hydrologiques*, *57*(6), 1118–1140. (WOS:000307640400007) doi: 10.1080/02626667.2012.701305

Wood, S. (2015). *A Brief History of Python.* Retrieved 2018-04-04, from https://hub.packtpub.com/brief-history-python/

Yim, S., Aing, C., Men, S., & Sovann, C. (2016, January). Applying PCSWMM for Stormwater Management in the Wat Phnom Sub Catchment, Phnom Penh, Cambodia. *Journal of Geography, Environment and Earth Science International*, *5*(3), 1–11. Retrieved 2018-03-29, from http://sciencedomain.org/abstract/13395 doi: 10.9734/JGEESI/2016/23525

Zaghloul, N., & Abu Kiefa, M. (2001, July). Neural network solution of inverse parameters used in the sensitivity-calibration analyses of the SWMM model simulations. *Advances in Engineering Software*, *32*(7), 587–595. Retrieved 2018-04-06, from http://linkinghub.elsevier.com/retrieve/pii/S0965997800000727 doi: 10.1016/S0965-9978(00)00072-7

Zhang, W., Li, T., & Dai, M. (2015, June). Uncertainty assessment of water quality modeling for a small-scale urban catchment using the GLUE methodology: a case study in Shanghai, China. *Environmental Science and Pollution Research*, *22*(12), 9241–9249. Retrieved 2018-03-07, from http://link.springer.com/10.1007/s11356-015-4085-7 doi: 10.1007/s11356-015-4085-7