LEVERAGING TECHNOLOGY TO ADD VALUE TO A PHASE II NPDES PERMIT

Marcus F. Aguilar

Thesis submitted to the faculty of the

Virginia Polytechnic Institute and State University

In partial fulfillment of the requirements for the degree of

Master of Science
In
Civil Engineering

Randel L. Dymond, Chair
Glenn E. Moglen
Kevin D. Young

May 1, 2013
Blacksburg, Virginia

Keywords: Stormwater, GIS, MS4, Hydrology, Programming, Urbanization

LEVERAGING TECHNOLOGY TO ADD VALUE TO A PHASE II NPDES PERMIT

Marcus F. Aguilar

ABSTRACT

In 1999, Phase II of the National Pollutant Discharge Elimination System engaged operators of small Municipal Separate Storm Sewer Systems (MS4) in the control of runoff from urban areas. The complex task of urban runoff mitigation has been investigated for several decades, resulting in a large variety of available computing and measurement tools for urban stormwater management. Unfortunately, these tools may not be available to the MS4 operator in a format that is both concise, and directly applicable. To address this need, this thesis recommends stormwater model creation and refinement strategies for Phase II MS4s using GIS and Python scripting. Further recommendations on using a popular discharge measurement technique for model calibration are provided. This workflow is then demonstrated in a watershed in Blacksburg, Virginia, where a unique MS4 permitting partnership allowed the development of these tools. Finally, further improvements to the workflow are suggested along with ideas for additional research for stormwater management in Phase II MS4s.

ACKNOWLEDGEMENTS

How does one acknowledge everyone who contributes to a several year-long project? Impossible, I say. Ideas and encouragement came from so many places the past two years that I'll certainly forget someone, though I'd at least like to thank those that have had an immediate impact on my work.

Dr. Dymond, thank you for asking the right questions, and not giving me the answers to them. Thank you for not ever telling me what to do. I've learned much from your style of leadership.

Funding for this work was provided by the Town of Blacksburg, and special thanks are due to Kafi Howard, Katherine Smith, and Matt Stolte for your willingness to answer my questions and provide insight.

Dan, Mark, Katelyn, Will, Kathleen, and David – thank you for being my Blacksburg family.

Thank you Lord, for the ability to read, write, and learn; and for the great joy that it brings me.

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

1. INTRODUCTION

1.1. BACKGROUND

There is a large and ever growing assortment of tools for simulating and predicting the diffusion of natural or man-made pollutants from the land surface, as it is now thought to be the leading cause of water quality problems in the United States (USEPA, 2002). The need to model these processes increased in 1999, as around 6,000 operators of Municipal Separate Storm Sewer Systems (MS4s) became responsible for controlling the quantity and quality of stormwater effluent under Phase II of the National Pollutant Discharge Elimination System (NPDES). Phase II regulations were notably different from Phase I in that the operator (i.e. owner of the storm sewer system) need not meet a numerical standard for compliance, but rather must implement certain "minimum control measures" (MCM) thought to reduce environmental damage in receiving waters (USEPA, 1999). The distinctly qualitative guidelines of this new regulation allowed flexibility in compliance, but also meant that stormwater managers would need to decide on a tool (i.e. rainfall/runoff model) to predict hydrologic response in their watersheds. The manager should search for a methodology that (Adapted from Pitt et al., 1999):

- Focuses on development at the tens of acres scale
- Is robust and flexible
- Is cognizant of the expense of data collection and management
- Is reproducible and consistent
- Uses the power of the desktop computer
- Uses appropriate levels of spatial and temporal discretization
- Accounts for uncertainty in the real and modeled system

In addition to these guidelines, the operator will need to evaluate randomness, and spatial and temporal variability (Chow et al., 1988) in the physical system. Many of the currently available models use GIS as a framework, as the benefits of doing so have been well documented (Huber and Strecker, 2008; Ogden et al., 2001; Seth, 2006). GIS offers a suitable structure for watershed conceptualization, and the literature is full of examples of modeling exercises in GIS. However, while the use of GIS allows for superior organization, visualization, and interpretation of watershed data, it also provides two methods of data storage – vector and raster, further increasing the number of available options for watershed modeling.

The difficult task of selecting and applying a modeling structure for stormwater management has presumably been performed for each MS4, as the now decade old program requires the control of post-construction runoff (USEPA, 1999). To this end, The Town of Blacksburg, Virginia and Virginia Tech created a unique relationship for MS4 compliance and stormwater research. This partnership provides the Town with the most current ideas on stormwater management from the literature, while Virginia Tech benefits from the variety of research opportunities in urban development and stormwater management (James and Dymond, 2012). The result of this ongoing relationship has also been a compilation and refinement of data structures, tools, and modeling techniques tailored to the needs of a small MS4.

1.2. Problem Statement

The abstract requirements of Phase II of the NPDES present a unique stormwater management challenge because of the small size and limited resources of the watersheds to which they apply. The EPA's commission of recreating pre-development hydrology was not paired with best practices for how to accomplish this, so local and regional authorities complied as they saw fit. There may, in fact, not be a single "best" practice, since Phase II includes around 6,000 communities in the U.S. with unique needs and funding situations; but certainly there are "better" practices. This paper attempts to convey these practices, answering the question, "How does a small MS4 leverage current technology to add the most value to its stormwater management program?"

1.3. Research Objectives

The research objectives for this project are to:

1. Present a GIS data structure and model suited to the small MS4 operator, and evaluate against other options.
2. Impart an efficient workflow supported by Python scripts to efficiently create and parameterize the model from Objective 1.
3. Comment on the appropriate use of acoustic Doppler discharge measurements for model calibration and refinement.
4. Present this procedure in a sample watershed
5. Identify weaknesses in the presented workflow and suggest areas for further research and improvement.

2. LITERATURE REVIEW

2.1. NPDES AND HISTORY OF COMPLIANCE

Water regulation in the United States has expanded in scope since the creation of the *Clean Water Act* (1972) to include stormwater runoff as a regulated source of water pollution. In 1990, the Environmental Protection Agency (EPA) ascribed stormwater pollution to the owners of subsurface storm drain systems, and so began the process now known as Municipal Separate Storm Sewer System (MS4) Permitting. This program was deployed in two phases; the first regulated urban areas of population 100,000 or larger, and was based on quantifiable improvements to water quality. Phase II of the MS4 program began in 1999, and governed urbanized areas (as defined by the U.S. Census) that were not already regulated under Phase I, as well as construction sites larger than 1 acre. But the Phase II regulations are distinctly different from Phase I regulations in that the operator need not meet a performance standard for compliance, but must instead implement certain "minimum control measures" (MCMs) thought to reduce environmental damage in receiving bodies (USEPA, 1999).

As the Phase II permit relies on technology-based standards, and as the mandate is unfunded, it has been suggested that most operators will meet only minimum requirements, and that minimum compliance may not actually improve water quality (Cave et al., 1999; Shamsi, 1996; S. S. White and Boswell, 2006). To address this concern, the EPA and the Center for Watershed Protection (CWP) created guidance manuals that provide further insight into effective post-construction runoff control programs (CWP, 2008; USEPA, 2010). These manuals expand on the objective of pre-development hydrology and provide some hydrologic methods (i.e. Rational, Natural Resources Conservation Service Curve Number), but the application of these methods is left to be determined by the stormwater manager. Without external guidance, the stormwater manager may turn to prior modeling experience for application, leaving a wealth of experiential knowledge from the literature and other small MS4s untapped.

2.2. THE VECTOR DATA STRUCTURE FOR STORMWATER MODELING AND MANAGEMENT

Literature presents an overwhelming selection of available hydrologic models as shown in Borah and Bera, (2004); Elliot and Trowsdale, (2007); Huber and Strecker, (2008); and Singh and Woolhiser, (2002). Choosing a model from these lists generally requires a categorization of modeling needs based on the following (after Chow et al., 1988):

1. Randomness in the model (stochastic or deterministic)
2. Temporal variability (steady or unsteady flow)
3. Spatial variability (lumped or distributed)

GIS has been employed in various ways to support the above modeling combinations as it provides two different methods for data storage: vector or raster (grid) format (Maidment, 1996). Therefore, along with the initial causal, temporal, and spatial categorization, the modeler must then determine the most suitable data structure to support the selected model.

Vector representations of a watershed lump regions of parameter homogeneity to define sub-watersheds, which are then attributed to a certain inflow location in a drainage network and routed to the watershed outfall. The vector model defines this lumped area as a polygon, and the hydraulic network as polylines connected to points. Early examples of vector based models can be found in Djokic and Maidment (1991) and Maidment (2002). A highly discretized vector model called the Tokyo Storm Runoff (TSR) model was used in Amaguchi et al. (2012) and Olsson et al. (2012) to assess flooding and impacts of climate change in urban areas based on highly explicit hydraulic definitions. The Urban Runoff Branching Structure Model (URBS-MO) has also been developed as a vector model that takes advantage of highly detailed morphological databases of urban areas called “urban databanks” (Rodriguez et al., 2008). Both of these models produced promising validation results.

This project does not intend to reproduce such a high level of discretization in modeling, but rather use highly detailed data to inform the parameterization process. Storm sewer flow, for example, has been shown to have an important role on urban response times (Choi et al., 2011; Dewals et al., 2012), suggesting the necessity of incorporating the pipe network in a model. Since the vector structure can explicitly define linkage between discrete objects, and because it allows for attribute indexing (Garbrecht et al., 2001; Johnson, 2009; Martin et al., 2005), it seems reasonable to argue that the vector structure is more capable of representing a small urban watershed where pipe flow constitutes a large component of the runoff process. Gironás et al. (2010) support this argument, as they use an elevation model supported by a vector pipe network as the benchmark by which to calibrate other models, stating that it is likely to be the most representative of actual runoff conditions in the watershed.

Another important component in a hydrologic model is the method used to model land surface elevation (Li and Wong, 2010). Elevation models in vector format are called Triangulated Irregular Networks (TINs), and are based on continuous interpolation between input elevation points (DeVantier and Feldman, 1993). The benefit of the TIN surface is that point elevation data is used in its entirety, while the raster format must sample the data at square increments. Li and Wong (2010) found that sampling of high resolution data can lead to inaccurate surface representations, leading to a poor depiction of hydrologic characteristics. The TIN surface, however, has the capacity to preserve all input data, and Jones et al. (1990) determined that TIN surfaces are suitable for hydrologic applications.

2.3. High Level Computing in Environmental Modeling

GIS was not originally conceived as a model in its own right, but rather a method of spatial database management. The recognized usefulness of GIS for environmental modeling led to the first attempts to integrate the two systems in the late 1980's (for a thorough history of GIS and hydrologic model integration, see Martin et al. (2005) or Sui and Maggio (1999). These efforts were an attempt to apply the large amount of spatial data required for hydrologic analysis to the temporal computing abilities of hydrologic models. It was soon discovered that this data manipulation could be done effectively using programming methods; an early example being Haddock and Jankowski's (1993) integration of PC-ARC/INFO with the Agricultural Non-Point Source model using Pascal. Although this is not a high level language, it shows the need for computing within GIS, and at its interface with the hydrologic model.

Karssenberg (2002) promotes the use of environmental modeling languages (EMLs) for hydrologists as they improve functionality without a large amount of computer science experience. These languages are at the highest level of computing (higher, even, than scripting), and while they are effective at iteration and error reduction, are worse in generic application. The EML uses hydrologic objects connected logically to provide a simple programming framework. Karssenberg suggests that EMLs may operate at too high a level for emerging models, and that languages closer to the system level may be in the future of hydrologic modeling.

As scripting has the low-level functionality of a system language, but the simple syntax of an EML, it has been recognized as a transformational tool for assessment and modeling in applied sciences (Bryan, 2013). A notable example of the usefulness of scripting in hydrologic modeling

is the integration of a processing engine in the ArcHydro data model (Whiteaker et al., 2006). This project extended GIS in the temporal domain using a schematic processor as applied through the ModelBuilder environment. Van Der Knijff et al. (2010) used Python as a wrapper for the PCRaster application, eliminating some of the restrictions inherent with the software. They found that this improved the input/output process without affecting the PCRaster source code. Abdella and Alfredsen (2010) used Python scripting to convert binary precipitation data into a standard raster format and projection, adjust the raster for discrepancies, and provide goodness-of-fit information based on comparison to rain gauge data. They found the use of Python's modules intuitive, and the language more flexible than Arc Macro Language (AML).

2.4. Flow Measurements, Model Refinement, and Model Calibration

The efficient creation of a simulation model for long term watershed management has little value unless the model's output can be compared with observed measurements in the natural system. In spite of this, stream and storm sewer gauging programs have had difficulty gaining traction with resource allocators, leading to a lack (and even deactivation) of gauges nationwide (Ogden et al., 2001). Without observations by which a model can be calibrated, even the most thoughtfully devised computer model will have only a theoretical ability to investigate and manage runoff response. Flow measurement, along with an understanding of how to appropriately interpret and use them for calibration, is therefore a necessary part of stormwater management (Harmel and Smith, 2007; Pitt et al., 1999).

Revision of input parameters to a hydrologic model in an attempt to improve the model's ability to simulate measured discharge data has been questioned in the literature from two directions. First, the idea that parameters supposedly representing physical reality can be adjusted to improve goodness-of-fit has been contested on the grounds that a model must provide good output for the right reasons (Ball, 2009; Beven, 2008; Kirchner, 2006; Klemeš, 1986a, 1986b). Second, measured discharge has been used for calibration without a suitable understanding of the measurement's characteristic uncertainty (Gupta et al., 1998; Harmel et al., 2006; Harmel et al., 2009).

The discharge measurement device used in this project was the Teledyne ISCO 2150 Area-Velocity Flowmeter (ISCO, 2011), and as such, the following review will pertain to the application of similar acoustic Doppler velocimeters (ADVs). These devices use a pressure

transducer to calculate water depth from head pressure. They have a second transducer that produces an ultrasonic beam which employs the Doppler shift effect to measure fluid velocity. Based on a known stage-area relationship, a stage measurement, and a velocity measurement, the processing component of the device calculates discharge. A thorough description of the ADV technology can be found in (Larrarte et al., 2008).

The manufacturer's stated accuracy of this device for velocity is ±0.1 ft/s (from -5 to +5 ft/s) and ±2% of reading (from 5 to 20 ft/s). For depth measurement, stated accuracy is ±0.010 ft (up to 34 ft.) (ISCO, 2011). Vermeyen (2004) tested two similar devices in a laboratory flume and reported errors ranging from -8.3% to 34.3% of benchmark discharges. Rehmel (2008) tested another similar device against the Price AA and Pygmy meters at 43 different USGS gauging stations and found that the device measures discharge within 5% of the Price meters. Finally, Bonakdari and Zinatizadeh (2011) determined that the location of a similar ADV in a large sewer pipe can produce measurements varying from 80 to 109% of actual mean velocity in the pipe.

An understanding of the uncertainty in ADV measurements should lead to a thorough inspection and refinement of the data itself as in Goring and Nikora (2002). This filtered data can then be used for model calibration as discussed in Harmel and Smith (2007). This paper applies the probable error range as described in Harmel et al. (2006) to incorporate measurement uncertainty in model performance assessment. The actual parameter tuning procedure varies based on the selected model, but a general methodology can be found in Bingeman et al. (2006).

2.5. Summary of Literature

Operators of small MS4s have been given the responsibility of preventing environmental degradation in natural water bodies without the benefit of external funding or access to a practical, condensed evaluation of recent innovations in stormwater management. These innovations include the consummate application of GIS for specific modeling needs, as well as the use of scripting for model creation and parameterization. The literature also provides technical insight on appropriate means and application of discharge measurement for stormwater model calibration. Availing this suite of directly relevant experiential knowledge to an operator could have important benefits on their ability to model and manage stormwater.

3. Leveraging Technology to Add Value to a Phase II NPDES Permit

3.1. Introduction

Urbanization has been shown in the past several decades to have a myriad of damaging effects on the quality of receiving waterways (Leopold, 1968), and management paradigms have evolved from the initial objective of flood prevention, to more recently, pollutant load reduction (Burns et al., 2012) and preservation of pre-development hydrology (Fletcher et al., 2012). Under Phase II of the NPDES, the onus of mitigating non-point source pollution was placed on the owners and operators of Phase II MS4s nationwide (USEPA, 1999). These regulations required approximately 6,000 small urban areas (UA) to devise and implement a program that would place restrictions on the discharge of runoff and the pollutants carried therein, in an attempt to preserve pre-development hydrology. The specific method for accomplishing this was intentionally left at the discretion of the operator, as the new regulations defined six generalized stormwater Minimum Control Measures (MCMs) in lieu of specific criteria. While allowing the creation and implementation of individualized stormwater management plans, this system also meant that each operator would need to create his or her own program for preventing illicit discharges and mitigating the effects of development through best management practices (BMPs) as specified by MCMs 3 and 5 respectively. The remaining four MCMs are educational in nature, or pertain to construction practices, and are not mentioned further in this paper.

Many methods for stormwater management and illicit discharge detection exist, however one of the most common approaches is through mathematical watershed modeling (Borah and Bera, 2003, 2004), providing the ability to perform large-scale hydrologic analysis. The abundance of available models has provided a basis for research on a variety of watersheds, but it has also lead to confusion and even misapplication (Klemeš, 1986b). The Phase II MS4 operator may be especially susceptible to this, as small urban areas have smaller resource pools and experience bases to solicit for guidance. Generalized criteria coupled with a lack of resources to aid in stormwater management oversight and enforcement has been shown to result in low to moderate compliance (S. S. White and Boswell, 2006). This minimal compliance may not be effectively improving hydrology, as Pitt et al. (1999) suggest that only a

comprehensive wet weather design methodology that investigates cumulative impacts of micro-development in a robust framework that is cost-effective, scale appropriate, and reproducible will be successful. Pitt and Clark (2008) state that the lag in implementation of proven technology is one of the biggest impediments to effective watershed management, and Miller et al. (2007) describe the need for tools for repeatable, accurate, and straightforward parameterization.

The auxiliary value of carefully structuring and defining a stormwater dataset is that it would also serve as an infrastructure asset management tool. Halfawy (2008) discusses the importance of an integrated dataset capable of both performance modeling and asset management tasks (such as condition assessment and asset prioritization) to prevent data fragmentation and further optimize resource use. Halfawy (2008) also discusses the need to formalize explicit process models to integrate data and software tools within and across municipal departments. This multi-purpose database concept is necessary because municipal authorities are perpetually required to evaluate, maintain and renew failing infrastructure with limited funds (Vanier, 2001). The array of decision support capabilities that a database will provide in completion is limited by the diligence in planning at conception, as Clark (1998) states that the database is the critical technology for both hydrologic modeling and asset management.

The purpose of this paper is to provide front-end guidance for small MS4s on building a data framework for stormwater management, and efficiently parameterizing a hydrologic model using GIS and the Python programming language. This is followed by a discussion on current technology for stormwater discharge measurement in urban watersheds and recommendations on how to appropriately interpret these measurements for model calibration and validation. This paper intentionally omits any specific suggestions concerning actual model selection, but rather provides an architecture that a selected model should accommodate in the context of small urban watersheds.

3.2. Technological Innovations in Modeling Stormwater Runoff

3.2.1. Defining an Enduring Data Framework and Workflow

The database and ensuing model of an urban watershed should be devised so that present and future needs can be accommodated without compromising workflow. This

is especially true for the operator of an MS4, because the chosen management model will need to be immediately useful, but also useful in the long-run. The format that is chosen at the front-end of a modeling project has critical consequences on the type of decision support it will be able to provide an operator. It is therefore imperative that initial watershed conceptualization take into account the anticipated role of the model in stormwater management.

Representing the urban watershed in a vector network structure allows the operator to explicitly identify variations in landform, and how they are topologically connected to receiving waters. If land disturbance is proposed, the operator can immediately identify the contiguous storm sewer system (and receiving water). The vector structure also allows the explicit analysis of the mitigation effects of a proposed management practice at the site scale, and the watershed scale. Examples of such highly distributed vector models are rare in the literature, as the amount of input data required increases rapidly with watershed size. However, for a small urban area, it may be reasonable to collect this level of detailed information, as illustrated in Amaguchi et al. (2012); James and Dymond (2012); and Olsson et al. (2012). Furthermore, it has been shown that the ability to analyze the effects of a combination of BMPs in a watershed is important, as systemic consequences of practices in series or parallel may be null or even negative (Emerson et al., 2005). It is therefore imperative that for planning decisions, the operator be able to isolate hydrologic and hydraulic conditions in specific subwatersheds and storm sewer structures, a feature of the vector network structure.

The vector model is comprised of three separate components in the GIS. The first is the representation of features on a Cartesian plane providing a geometric representation of the watershed. Subwatersheds and ponds are defined as polygons, conveyances as polylines, and nodes (i.e. catch basins, manholes, and pond outlet structures) as points. The geometry of each feature is defined based on coordinates, and features are related to each other through spatial relationships. Conveyances, for example, must begin and terminate at a node, and each subwatershed must contain one outflow node. Feature geometry and spatial relationships are the basis for network topology, but it is also noteworthy that feature geometry can be accessed and revised through geoprocessing tools and/or custom programming code. The ability to access and modify feature

geometry is an important advantage of the vector structure, as spatial manipulation can be easily automated.

The second component of the vector model is the attribute data that are associated with each geometric feature. The two are related based on a unique identifier such that each feature maintains its attributes even when its geometry is edited. The ability to relate an attribute table to discrete geometric features is perhaps the largest advantage that the vector structure maintains over the raster structure. Detailed information about each feature is stored in a series of fields, which must be carefully selected at the front-end of the project. Fields should allow for a thorough description of the feature, but too many fields can complicate data querying. For example, a pipe feature class should contain information about size, material, and connectivity. It should not, however contain fields describing attributes of the upstream and downstream nodes; this should be placed in a separate point feature class.

The third component of the vector structure is the triangulated irregular network (TIN) surface (Figure 1), which provides the model with necessary elevation information. TINs connect input point elevation features to create a continuous network of triangular surfaces. Points connected continuously by irregular triangulation allow a spot elevation to be extracted from the triangles at any point on the surface. By contrast, the grid structure attempts to define generally irregular elevation measurements (from LiDAR or field survey) using a regular matrix, which requires some type of data sampling. It is especially true in urban areas, where the built environment causes surface discontinuities (i.e. curbs, buildings, retaining walls), that elevation measurements be preserved and interpreted in a manner that can account for such abrupt irregularity. Hydrologic model results depend heavily on the quality of the digital elevation model (Li and Wong, 2010; Moglen and Hartman, 2001), and this workflow is no exception. Due to difficulties in the identification of LiDAR and TIN errors, irregularities in the surface may occur, which would lead to non-representative model results. Fortunately, LiDAR has been shown to be more reliable for the extraction of hydrologic information than traditional field survey methods (Wang and Yu, 2012), and LiDAR accuracy has been reported between 17 and 26 cm for flat areas, and twice that for slopes steeper than 25 degrees (Hodgson and Bresnahan, 2004).

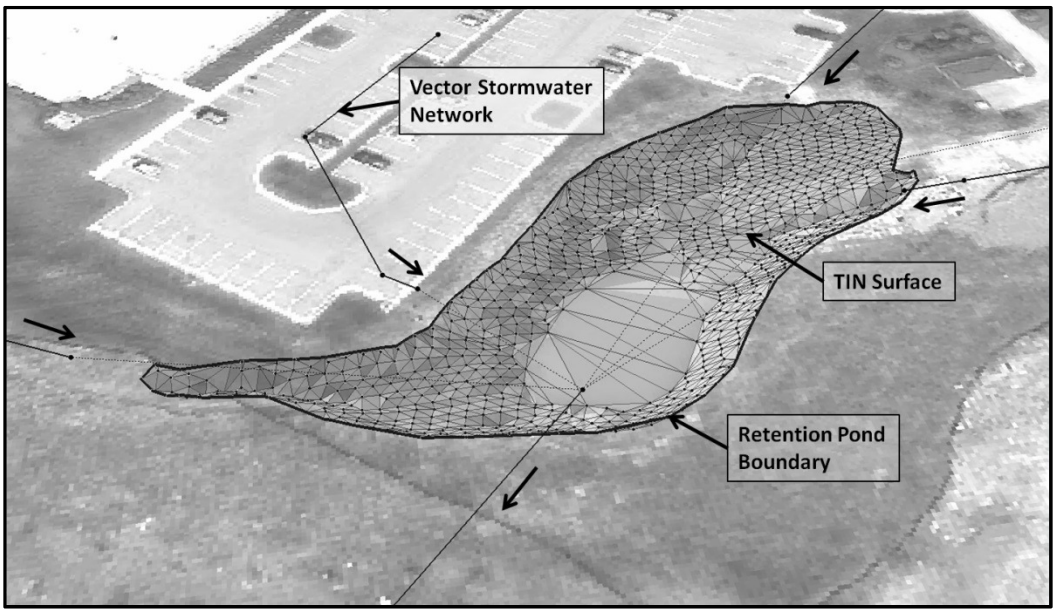


Figure 1 - Stormwater Retention Pond Defined by Pipe Vectors and TIN Surface

The coupling of tabular attributes with spatial locations allows complex queries that leverage both components of the data. The time required to build and maintain such a database is certainly greater than other options, but the value of a comprehensive stormwater information system will be realized in both hydrologic decision making and infrastructure asset management. The use of this information system for asset management is beyond the scope of this paper, but initial decisions concerning dataset, feature class, and field structure in a stormwater database should consider asset management needs as well. Concerning the creation of this database, it will be shown that the personnel and resource demands of data collection and assimilation, which may prevent an operator from creating such a database, can be greatly reduced using GIS and Python.

3.2.2. PYTHON SCRIPTING IN GIS FOR MODEL PARAMETERIZATION

Based on the stated resource limitations of a small MS4 operator, it may seem unreasonable to include programming as a suggested tool; however the perceived difficulties of learning a programming language are alleviated by the simple syntax and high level operation of Python (Lutz, 1996). Scripting is recommended because of three fundamental features of the Python language for model development in GIS:

1. Any geoprocessing tool can be written in command line format and therefore automated in sequences, loops, and if/else statements. This minimizes the need for input arguments thereby minimizing the possibility of user error. This also allows for standardized workflow and greatly reduces the need to repeat common computations.
2. Python can read and write feature geometry, a critical function for vector network creation.
3. Scripts can extend beyond the GIS environment, using system functions or other applications for support. This is useful for data formatting and output.

This section describes how a stormwater manager could leverage each of these features, and provides examples of each in pseudocode designed to encourage individual modification and application. While this paper discusses Python specifically for use by a small MS4 operator or stormwater manager, the language has other advantages, as noted in Bryan (2012). All the following scripts were created in Python 2.7 (VanRossum, 2013), and applied using ArcGIS 10.1 (ESRI, 2013).

Figure 2 represents an example workflow for processing input data for the creation of a stormwater geodatabase that can be used as the input for a stormwater model. The first notable trait of the diagram is how few processes are involved, when in reality the seven scripts comprise 78 functions organized into tools useful to a stormwater manager. The scripts are described in this paper based on the major concepts they use to improve workflow; descriptions of the 'Sewer Topology Creator' and 'Extract and Filter Ground Points' were omitted because they do not add any further knowledge beyond what is already presented.

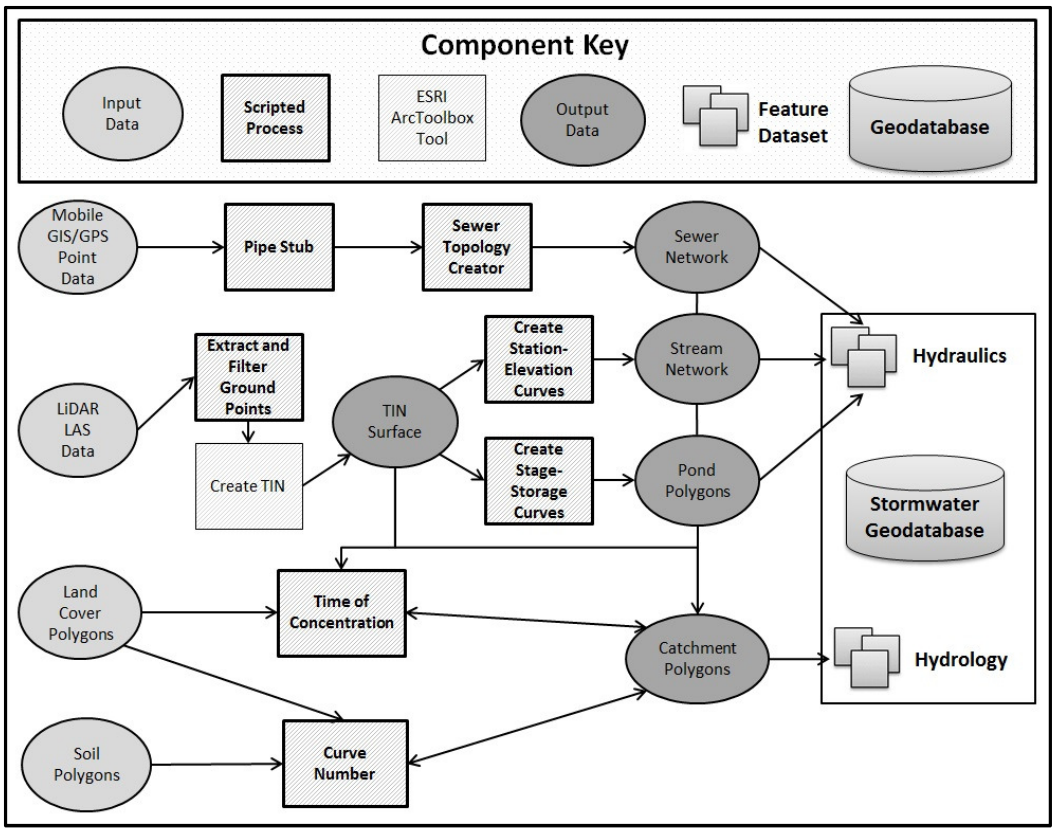


Figure 2 - Stormwater Workflow in GIS

An example of the usefulness of scripting for automation is the "Curve Number" script as shown in Figure 3. This script requires only a basic understanding of how to operate Python, but condenses 22 processes in a single tool. The four lines in Figure 3 describe the most important logical steps in the script, but omit other formatting and calculation steps. This shows the immediate benefit of automation, especially when the delineation of subbasins changes with different management scenarios.

```
Intersect(subbasins, soils, landcover)
RelationalJoin(intersect, lookup_table)
FieldCalculation(intersect.CN, lookup_table.CN)
Dissolve(intersect)
```

Figure 3 - Curve Number Script Pseudocode

Although process consolidation is simple and has immediate benefits, learning to use Python's iterators and logic statements results in a significant increase in productivity, as repetitive processes can be completed using loops. A tool was created (Figure 4) that

uses a process nested in a 'for' loop to extract a pond stage-storage curve from a TIN surface, showing the usefulness of the iterator in Python. The script then formats the text in lists, and writes them to a .csv file (not shown in Figure 4) that can be used as needed for modeling, eliminating repetitive processes in storage volume estimation. There are a variety of practical applications for the Python iterator in GIS, but generally it is used to perform work on a collection of geometries, fields, rows, or feature classes. In the stage-storage script, the elevation of each contour feature (stored in a row) is extracted and used as the input for the 'Calculate_Vol_Below' process.

```
CreateContours(TIN_Surface, Elevation_Interval)
for contour in contours:
      Calculate_Vol_Below(TIN_Surface, contour)
```

Figure 4 - Stage - Storage Script Pseudocode

Certain other applications may require that a script make simple decisions concerning which process to run, or which calculation to use based on preconditions; this can be effectively done using 'if/elif/else' logic. This logic was used for the estimation of time of concentration (Figure 5), as water will encounter a variety of land covers and flow conditions before it reaches a subbasin outlet. The usefulness of the vector storm sewer system should be noted here, as the script must determine when to perform channel flow calculations. The code shown in Figure 5 requires user input drainage paths through an urban watershed, and begins by converting this polyline to a 3D polyline. Flowpaths are then broken into components by intersecting with land cover and pipe network features followed by a series of conditional statements for time calculation. Components are finally merged into a single line using the "Dissolve" tool. The output of this tool is the user defined flowpath with an associated time value in the attribute table. Optionally a user could define multiple flow paths for a single area, and evaluate the array of resulting time outputs. Assembling geoprocesses, conditional statements, and calculations into a script reduces work, documents workflow, and allows the operator to select from a range of values. The initial time investment required to create such a script may be large, but the result is a consistent, repeatable process that is easy to use, and provides a range of possible time of concentration values for a given area. The same is true of other scripts related in this paper, and it

should be noted that the first steps of translating a GIS workflow into a script tend to be the most difficult.

```
Interpolate_Shape(TIN_Surface, Flowpaths) #Write slope and
      elevation data to flowpaths
Intersect(Flowpath, Pipes, Land_Cover)
if Flowpaths == Pipe_Flow:
      #Do Pipe Flow Calculations
else:
      if Flowpaths == 'Sheet':
            if Land_Cover == 'Paved':
                  #Calculate Paved Sheet Flow time as f(slope)
            else:
                  #Calculate Unpaved Sheet Flow time as f(slope)
      elif Flowpaths == 'Shallow':
            #Perform calculations for shallow concentrated flow
Dissolve(Flowpaths)
```

Figure 5 - Time of Concentration Script Pseudocode

Automating geoprocesses is useful for relationships between features or feature classes, but an operation that requires access to, and manipulation of the spatial component of a feature requires a separate method. This spatial information is located in the 'Shape' field of an attribute table, and can be accessed through Python's '.shape' method. The Pipe Stub script (Figure 6) is an example of the benefit of this ability, as azimuths stored in point features are used to create line features representing pipes. The tool also allows pipe attribute information (such as size and material) to be transferred from the node feature class in which it is collected, to its respective line feature. This means that a comprehensive stormwater network model could be created with only minimal field survey resources.

```
#Create a 'for' loop through each node in the node feature class, then
#nest a 'for' loop through all the fields for each feature
if field.name == number_of_conveyances:
      num_conv = row.getValue(field.name) #determine number of pipes
      #connecting to node from attribute table
            for i in range(1, num_conv+1): #loop through pipes
                  if field.name == "Azimuth"+str(i): #find Azimuth_i
                        azimuth = row.getValue(field.name) #read Azimuth_i
                        #pipe starting coords.
                        [x1,y1] = node.X, node.Y
                        #pipe ending coords. where 'StubLength' is the
                        #desired length of the pipe stub
                        x2 = x1+StubLength*math.sin(azimuth)
                        y2 = y1+StubLength*math.cos(azimuth)
                        #use insert cursor here with 'feature.shape'
                        #method to write beginning and ending coordinates
                        #to line feature
                  #find other pipe attributes from node feature
                  if field.name == "Other_Attribute"+str(i):
                        #write attribute to pipe feature
                        feature.OtherAtt=row.getValue(field.name)
```

Figure 6 - Pipe Stub Script Pseudocode

Data management requirements increase with watershed size and model scale, and the ability to read, write, interpret, and reformat this data en masse can considerably improve interoperability. For this purpose, Python can be employed as a mechanism for data exchange between otherwise incompatible sources. For example, a cross section script may extract station-elevation data from a TIN surface using absolute elevations, while the modeling software may need elevations relative to the cross section invert (minimum elevation in the cross section). After the data are extracted and written to a .csv file, Python can read and rewrite the file to the necessary format by importing Python's 'csv' site package (Figure 7). This same manipulation could be completed using spreadsheet equations, however it is advantageous to package the entire cross section workflow in a single script that does not require intermediate user input.

```
import csv #import Python's csv methods
excel_file = cross_sections_raw.csv #input csv filepath
outfile = cross_sections_adj.csv #output csv filepath
#create empty lists to store data
rows = []
rows_adj = []
with open(excel_file, 'rb') as csvinput: #open unformatted file
    with open(outfile, 'wb') as csvoutput: #create formatted file
        reader = csv.reader(csvinput, delimiter = ',')
        writer = csv.writer(csvoutput, lineterminator = '\n')
        reader.next() #skip header in reader
             for i, line in enumerate(reader): #iterate and count
                   rows.append((i,int(line[0]),line[1],line[2]))
                   #Write counter, ID, station, and elevation to list
       #Use Python's if/elif/else logic here to change datum in
       #cross sections to 0; write to 'rows_adj'
       writer.writerows(rows_adj) #write 'rows_adj' to new file
csvinput.close() #close input file
csvoutput.close() #close output file
```

Figure 7 - Station - Elevation Pseudocode

The disadvantage to the operator of investing in a specific language, is that each language has a limited life before it is deprecated. Fortunately, when this occurs, the concepts and code structure can be reused because they are language independent, although syntax will need to be modified. This problem is true of adopting any software language, and emphasizes the need for readable code, and in-line documentation.

3.2.3. Model Creation, Refinement, and Execution

Implementing process automation in a formalized stormwater workflow imparts consistency in a field that is otherwise highly susceptible to subjectivity and error. Even with rigorous automation, however, it is possible that once the data is used for modeling, problems with the physical representation of the watershed will arise. Examples of this include negative pipe slopes or the misrepresentation of a cross section as a result of LiDAR inaccuracy, leading to erroneous modeling results or even model failure. It is critical that refinements to the model (or database) do not physically mischaracterize the system, as a hydrologic model must work well for the right reasons (Kirchner, 2006). It is especially important to maintain database integrity if it is also to be used for asset management. At this stage of model creation, simulated output should not be compared to observations, as the immediate objective is only to assure

that the model is functioning appropriately with the given input data and modeling environment, and not that it matches observations.

It is critical that the user understand the computational method of runoff estimation employed by the model in order to properly troubleshoot calculation problems. If the causal relationship between model parameters and model results are known, the source of continuity and instability issues in models can be more readily identified, and the number of iterations of model adjustment reduced. Although variations may exist in reported information from model runs, most models should provide inflow and outflow volumes from which model continuity can be evaluated. If inflow is significantly greater than outflow, the hydraulic network is probably surcharged and runoff may be 'lost,' depending on how the model emulates flooding. The inverse result may suggest that there is a computational error concerning pond storage or release. Either way, the hydraulic system should be carefully scrutinized, with additional collection and refinement of field data if necessary. If possible, hydraulic conditions in the conveyance network should also be checked for abnormal flow conditions such as excessive velocities or super-critical flow. Depending on model selection and complexity, it may also be possible to detect calculation instabilities by visual inspection of the outflow hydrograph. Problems with instability may be a result of inappropriate calculation settings, such as time or distance calculation increments for non-steady-state models.

During the data refinement process, the user should grow comfortable with data exchange between the geodatabase and the model, as well as the format of model output. There are a variety of methods for coupling GIS with hydrologic models (Sui and Maggio, 1999), and the operator should be familiar with the interaction of the two during the model refinement stage. This will not only assure that data are not lost during database and model updates, but will also improve the operator's ability to calibrate and validate the model.

3.2.4. Application of Acoustic Doppler Measurements in Storm Sewer Systems for Model Calibration and Validation

The efficacy of a hydrologic model is determined by how well it can reproduce the response of the physical system through the synthesis of input data (Klemeš, 1986a). Stormwater management models can be calibrated to synthesize rainfall data to match

observed hydrographs or pollutographs, but only if a sufficient amount of this data have been collected for both calibration and validation. For this reason, it is imperative that continuous, well controlled measurements be collected as a component of a Phase II stormwater management program, even if they are not directly required by regulation.

The response to rainfall in a watershed depends on several geophysical processes which may be difficult to specifically characterize. A practical first step, however, is to monitor the quantity of water that passes the watershed outlet, as this provides baseflow, peak, time to peak, and volume information. It also allows the calculation of watershed abstractions through the balancing of volume, and can provide mass transport information if concomitant pollutant concentrations are recorded. Since there are certain methods for indirectly estimating nutrient loading (Schueler, 1987) and sediment transport (Yang, 1996) from discharge information, and because a small MS4 may not have the resources to implement water quality monitoring, constituent measurement should normally be secondary to quantity measurement for this application.

Initiating a discharge measurement protocol for an urban watershed should begin with the selection of an appropriate device, as well as a location that provides suitable hydraulic conditions. A flow measurement device for an urban watershed should:

- Function at a wide range of velocities and flow depths
- Withstand adverse flow conditions, such as heavy debris and sediment
- Maintain transparency in methods by which measurement and processing occurs
- Deliver measurements in a response time suited to the operator (i.e. a device for flood forecasting would need a much quicker response than a flume for agricultural runoff measurement).

The location that the sensor is placed should:

- Be relatively accessible for inspection, maintenance, and calibration
- Not be overly susceptible to vandalism or theft
- Not experience supercritical flow or a transition to supercritical flow during storm events
- Not be at or near pipe intersections to prevent bidirectional stream lines

The measurement device that has gained traction in urban environments is the Acoustic Doppler velocimeter (ADV), as it meets the above requirements and is autonomous in placement (Larrarte et al., 2008). Many of these sensors are capable of continuous measurement with communication systems capable of wirelessly transmitting data to a remote repository, and are small enough to be installed in configurations where a flume or weir would be impractical. They have a smaller profile than a constructed device and are therefore less susceptible to debris accumulation, and their small size also makes it possible to move them easily if necessary.

The concept of using discharge measurements from an ADV (or other device) as a benchmark for model calibration is that the model best replicates response when simulated results approach a perfect fit with these measurements. Implicit with this concept, however, is that there is zero uncertainty associated with the measurement, which is not true. The stormwater manager may therefore spend an unwarranted amount of time adjusting and re-parameterizing a model to fit data that are inherently uncertain. Harmel and Smith (2007) address this issue by developing the "Probable Error Range" (PER) for measurements. The ADV measures flow area (as a function of stage) and velocity, and therefore has two sources of measurement error. The PER for the *ith* measurement from an ADV is:

$$PER_i = \sqrt{E_{h,i}^2 + E_{v,i}^2} \quad (1)$$

Where $E_{h,i}$ = Error associated with the *ith* stage measurement (±%) and $E_{v,i}$ = Error associated with the *ith* velocity measurement (±%). Figure 8 describes these sources for a range of depths and velocities based on manufacturer specifications for the ISCO 2150 ADV (ISCO, 2011), and it is notable that low depth and velocity measurements result in higher uncertainty. It may be argued that this phenomenon is related to the presentation of uncertainty in absolute units; however uncertainty will increase as the ADV's sample volume decreases in proportion with total flow area (during low flow conditions). Since each measurement will result in different component errors, the PER should be calculated for the *ith* measurement, and then used to compute upper and lower bounds.

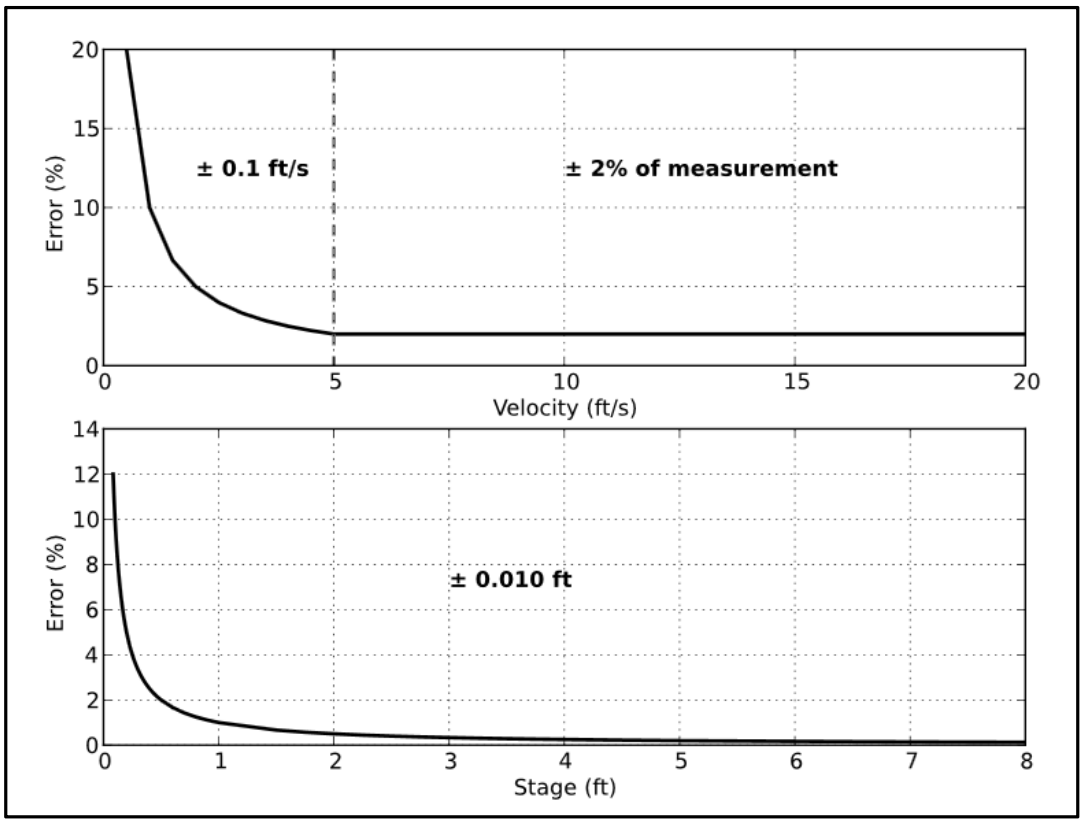


Figure 8 - Measurement Error as a Function of Velocity and Depth Based on Specifications in ISCO (2011)

Only after ascertaining the measurement uncertainty bounds should goodness-of-fit (GOF) calculations be performed based on Harmel et al. (2010) or Harmel and Smith (2007) if both model and measurement uncertainty are available. If the initial GOF values are not acceptable, some adjustment to the model will need to be made to improve these values. The highly constrained parameterization process as described in sections 3.2.1 and 3.2.2 means that the available parameters with which to adjust a model may be fewer. Directly quantifiable physical parameters such as pipe size or topology should not be adjusted; instead, the user will need to identify and re-evaluate abstractions, assumptions, and empirical equations in the workflow and their suitability for the specific modeling need.

The process of calibration itself can range from simple adjustment of one or two parameters, to a complex automated scheme that optimizes a model based on multiple objectives and constraints. A Phase II MS4 operator will probably not need such complex methods, but even simple calibration can be difficult. Generally, the operator will need to first prioritize the metrics by which the model will be judged. The EPA suggests pre-

development hydrology as an objective for managing urban stormwater (USEPA, 2010), so a model should be able to recreate the form of an observed hydrograph, as well as the peak and volume. Systematic form comparison can be measured through GOF metrics such as the index of agreement, Nash-Sutcliffe efficiency, or others described in Moriasi et al. (2007). The operator will need to keep a thorough record of model state changes and GOF results during this process, as it is likely that the model will need to change as future storm events become available. At the initiation of a discharge measurement program, insufficient data will be available to provide a wide range of storm events for calibration, especially for high return period storms. As these storms occur, the operator should be able to return to the initial calibration protocol and make changes as necessary. The operator may find that calibration becomes a compromise between metrics, or differing storm event types. This limitation should be viewed in light of model and measurement uncertainty as discussed previously. The lack of storm events and the necessity for compromise in calibration do not nullify the usefulness of the model, but instead define the limits by which management decisions can be made from the model output.

3.3. Application – The Town of Blacksburg, Virginia

Pursuant to the innovations presented in this paper is a working relationship with the Town of Blacksburg, Virginia created for the purpose of research and development in stormwater management and MS4 permitting. This relationship is unique because minimum regulatory compliance does not usually inspire research, so the opportunity for methodological advancement in this small subset of stormwater management is evident. The presented methods were applied to the 400 acre "Central Stroubles" watershed with 30.6% impervious area (calculation performed based on the 'Detailed' method and data in C. M. White (2011)). It is notable that this area constitutes the headwaters of a small upland stream, and is directly responsible for any hydrologic impacts at the watershed point of interest (POI).

The watershed was first delineated to the POI using a TIN surface, and then the hydraulic network was surveyed using a Trimble Juno handheld GPS device with ESRI's ArcPad software (ESRI, 2008). GPS location was reconciled with aerial photography loaded on the device, and point features were stored at all manholes, inlets, pond outlet structures, and endwalls. In addition, all contiguous pipe azimuths were stored for input into the Pipe Stub script. After the script was run, the pipe network was connected, and it became possible to visualize the storm

sewer system. Stream lines were digitized as necessary, and point features were placed at locations of geomorphic interest for station-elevation extraction using the Cross Section script. Additional field survey was performed until the topology of the watershed was well-defined in the GIS. Once the drainage network was created, subwatersheds were manually delineated based on the TIN surface, aerial photography, land cover, and pipe network. Although watershed delineation has been automated (Baker et al., 2006; Miller et al., 2007), automation has found only limited success in urban areas at the tens of acres scale (Gironás et al., 2009). The authors determined that it was beyond the scope of this project to design an automation procedure that accounts for both natural topography and built features at the 1 to 50 acre scale. It should be noted, however, that manual small scale delineation was only possible because the drainage network could be accurately visualized and because the TIN incorporated small surface discontinuities in contour generation.

After each subwatershed was delineated, the Curve Number and Time of Concentration scripts were used as shown in Figure 3 and Figure 5 to parameterize each area. These scripts not only provide the parameter itself, but also statistics from intermediate calculations, which can aid in identification of errors in delineation or flow paths. It is important to include results from intermediate calculations for quality assurance when automating tasks; otherwise the user may accept a value without the ability to judge whether that value is reasonable or not.

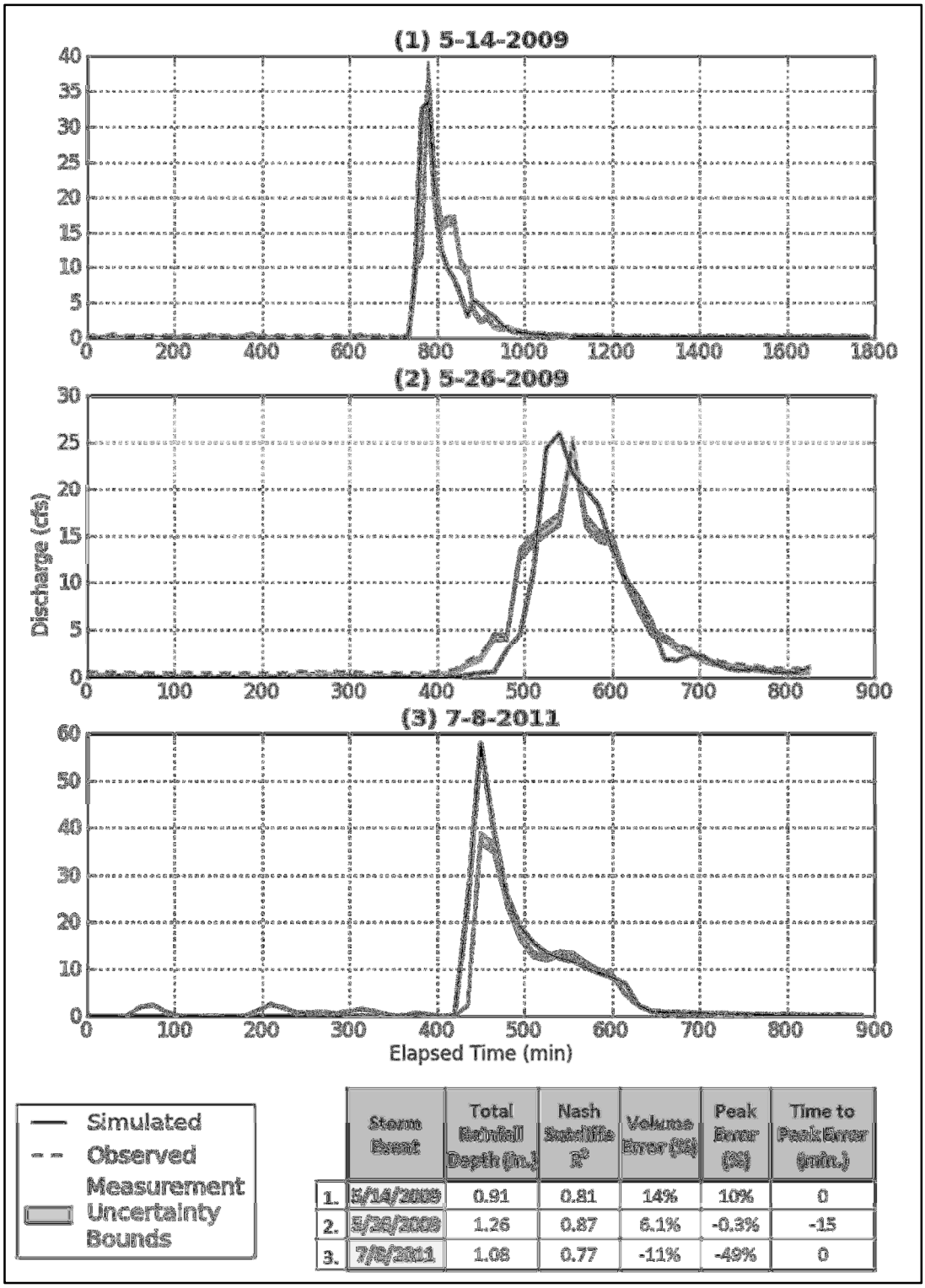
Once the pipe network and sub-catchments were well-defined, stream geometry and pond storage characteristics were determined. The watershed modeled in this project contains over 100 stream locations of geomorphic importance and 10 ponds, so it was beneficial to automate the station-elevation and stage-storage procedure as shown in Figure 4. Extraction of cross section and pond information was the concluding step for watershed definition, and it is at this point that the data were sufficient to serve as input to a hydrologic/hydraulic (H&H) model.

This study used SewerGEMs (Bentley Systems Inc., 2013) for modeling, which provides a convenient GIS data migration tool called ModelBuilder. This application connects to a data source, such as a spreadsheet, database, or geodatabase and reproduces the data in a format that can be interpreted by the model. Updates to the model or geodatabase are synchronized in both directions using the ModelBuilder, and results can be passed to the geodatabase for improved visualization in GIS. Understanding, developing, and refining this connection

required several iterations, but the eventual benefit was that data and results could be passed between the geodatabase and model without any intermediate manipulation.

The SewerGEMs model uses an implicit four-point, finite-difference numerical solution to the one-dimensional Saint-Venant equations to compute system hydraulics (Jin et al., 2002), and provides an array of methods to compute hydrology including the Natural Resource Conservation Service CN method (SCS, 1972), which was used in this project. The advantage to using the Saint-Venant equations for storm sewer modeling is that a hydrograph and hydraulic grade line solution can be produced at any point in the system, rather than just the system outlet. The disadvantage is that in highly complex systems, the solver may have difficulties converging on a solution. The refinement methods presented in section 2.3 were applied to improve model stability, and prepare the model for calibration.

Stormwater management regulations in Virginia are largely based on peak discharge and volume for discrete storm events, so the model calibration procedure was adjusted accordingly. Since it is also well known that the hydrograph shape has an important role in the health of a receiving stream (McCuen and Moglen, 1988), the Nash Sutcliffe R^2 value was also used to compare observed and simulated hydrographs (Nash and Sutcliffe, 1970). The calculation of these GOF parameters were modified for measurement uncertainty based on the idea presented in Harmel and Smith (2007) that a model should be calibrated to uncertainty bounds rather than a single value. Figure 9 presents an example of how this pairwise comparison appears in practice. This method generally suggests a more reconciled model than traditional methods, though Harmel and Smith (2007) point out that this method will not make a poor model appear satisfactory. In order to more completely account for uncertainty during model evaluation, uncertainties associated with the creation and operation of the model should also be considered, though discussion of model uncertainty is excluded from this paper as any model could be used with the presented workflow.



	Storm Event	Total Rainfall Depth (in.)	Nash Sutcliffe R^2	Volume Error (%)	Peak Error (%)	Time to Peak Error (min.)
1.	5/14/2009	0.91	0.81	14%	10%	0
2.	5/26/2009	1.26	0.87	6.1%	-0.3%	-15
3.	7/8/2011	1.08	0.77	-11%	-49%	0

Figure 9 - Central Stroubles Hydrographs (Including Measurement Uncertainty for ISCO 2150) and GOF Results

The GOF results in Figure 9 demonstrate the difficulty that an operator may encounter in calibrating a stormwater model. The same model that produced high GOF for storms (1) and (2) could not effectively reproduce the peak in storm (3), although the Nash-Sutcliffe R^2 for this storm is acceptable (i.e. greater than 0.75, from Moriasi et al. (2007)). Likewise, the event that had the overall best fit (2), had a 15 minute shift in time to peak (one reporting time interval). Furthermore, the storm events used for calibration are all summer storms, and smaller than 1 yr. events. These storms were used due to the lack of availability of additional storm events in the rainfall and/or runoff periods of record. This was caused partially by a lack of storm events over the project duration, but also due to inconsistencies in measurement recording. As a wider variety of rainfall event data become available, the operator will need to recalibrate the model based on these events to expand the precipitation conditions under which the model can be used.

The calibration procedure used for this work was the least automated of the major tasks involved in this project. This was because the SewerGEMs modeling software does not include a method for calibration, and because a suitable workaround could not be devised. Time to complete each major task is shown in Table 1, normalized by a 1000 acre area. This time is then calculated as a percentage of total modeling time to demonstrate the relative amount of time spent on each task.

Task	Man Days/1000 Acre	Percent of Total Effort
Field Data Collection	15	36%
Network Creation, Subwatershed Delineation	8	19%
Hydrologic Parameterization (CN, Tc)	1	3%
Hydraulic Parameterization (Stage-Storage, Station-Elevation)	1	3%
Model Construction, Refinement	5	11%
Model Calibration	13	29%
Total	43	100%

Table 1 - Normalized Time to Complete Major Tasks

3.4. DISCUSSION

The presented GOF metrics show that the procedures described in this paper for model creation and evaluation can support a well-functioning hydrologic model, and it is at the completion of this calibration exercise that the model reveals its worth as a decision support

tool for a Phase II MS4 operator. The steps leading up to the calibrated model as described in this paper are:

1. Conceptually define the watershed using a vector geodatabase and corresponding workflow. This enables explicit spatial and tabular representation of the system, and provides flexibility as the project matures.
2. Use Python's iterators, logic, spatial editing, and input/output capabilities to create and parameterize the stormwater geodatabase.
3. Apply the geodatabase to a modeling software of choice (this project used SewerGEMs), and refine the data based on preliminary model output.
4. Implement the acoustic Doppler velocimeter as a discharge measurement device. Ascertain the uncertainty associated with the device, and calibrate the model according to the error bounds associated with this uncertainty.

The end-product is an interactive environment which aids a stormwater manager in decision making and infrastructure resource allocation based on the hydrologic output of a set of development scenarios. The operator is faced with the daily task of reviewing development plans and the corresponding proposed mitigation practices, and this framework allows a review of the impacts of development at the site level, as well as at the watershed level independent of the developer's calculations. The form of the presented geodatabase allows the operator to identify specific locations in the storm sewer system that may require improvements to accommodate development, and also provides a platform for regional mitigation planning.

Funding acquisition for stormwater infrastructure improvements is normally part of the operator's role, and the structure presented will help the operator promote a better understanding of why a stormwater fee may be necessary. Storm sewer system maps as created through GIS, coupled with graphical output from a hydrologic model may improve public understanding of runoff processes, and explain why expensive stormwater improvements are necessary. A highly detailed characterization of a watershed as presented in this paper allows a resident to visualize the contribution their property is making to the storm sewer system, as well as the body of water that is eventually receiving that discharge.

Such detail may only be possible for a small MS4 if a watershed database can move from conception to completion without extensive time or resource use. Applying Python in GIS as a

process synthesizer is presented in this paper as a solution for assimilating large amounts of data with few resources, as demonstrated in Table 1. The 'Field Data Collection' task does not allow for automation by definition, and the 'Model Calibration' task was limited by the use of proprietary modeling software. Nearly two-thirds of the workflow was consumed by these two tasks, while the remaining four tasks, though large in scope, constituted only a small portion of the total time effort. An operator repeating this workflow, once data collection was completed, would be able to proceed rapidly through the steps leading up to a functional model, but would then be restricted by the intractable nature of manual calibration. The disparity in time-cost of manual over automated processes suggests the benefit of implementing scripting wherever possible in a workflow.

Lack of GIS or programming skills may impede the adoption of these methods, but if the tools are implemented for simplicity, the operator will not need to know how to use Python, but only how to use the tools. If the use of Python scripting is accepted by the MS4 community, future research may include the integration of a hydrologic model within the GIS interface using Python, similar to the study performed in Whiteaker et al. (2006). The advantage of this for municipal operations would be that the entire workflow from data collection to model results would be housed in a single software application without a need to convert or exchange data. This sort of embedded model may be a better solution for a municipal authority that values consistency and simplicity in workflow over a large variety of modeling options and features.

'Better' solutions for managing stormwater for a Phase II MS4 operator is the intent of this paper, as an all-encompassing 'best' solution is not likely to exist because of the diversity of the municipalities involved. Permitting authorities may find some benefit to auditing, quantifying, and rating compliance in their region (S. S. White and Boswell, 2006) to gain insight into the obstacles faced by their municipalities. If shortcomings in perception, culture, and technology can be identified, tools for improved management (such as those presented in this paper) can be developed and implemented to address patterns of non-compliance at a larger scale. These tools can help remove some of the 'art' from stormwater management (Pitt and Clark, 2008), thereby improving the Phase II MS4's ability to resourcefully improve water quality at the local scale.

4. Discussion and Conclusions

The modeling output characterized by GOF does not provide a holistic picture of the results of this study, as the innovations presented apply mostly to the front-end preparation and back-end refinement of a modeling curriculum. This paper does not present a new mathematical model of physical processes, or even a new application of a previously conceived model, so the GOF results are presented for the sake of posterity; to assure the reader that the disclosed ideas did, in fact, lead to acceptable modeling results. The more important discussion point is the application of the database structure, scripting tools, and discharge measurements as presented. Though it may be difficult to formally quantify and compare one workflow to another without rigorous time-cost analysis, the information provided in Table 1 presents the relative time differences between tasks in this project. Calibrating the Central Stroubles model required 29% of the total effort, and as calibration is an ongoing procedure, this number will increase by proportion as the project persists. Automation would facilitate this otherwise complex and time consuming process, enabling the rapid iterations that expedited other steps of the modeling process mentioned in this paper.

The product of these formalized and automated processes is a functional, flexible, and highly detailed characterization of a watershed that can be used by the operator of a Phase II MS4 in several manners. Any alteration (or proposed alteration) to the watershed can be inspected for the impact that it makes to the hydrologic regime at the watershed POI, or at any other point in the system. The explicit nature of the vector data structure allows the identification of insufficiencies in the storm sewer network, improving allocation of infrastructure renewal resources. The ability to specifically correlate a development project to storm sewer failure, and convey this causality using the visualization capabilities of GIS could also have an important role in the administration of stormwater fees. These fees have experienced public resistance because they are seen as unfair and unnecessary, especially in the midst of a poor economic climate (Ellard, 2010), but a storm sewer system model that relates flooding or water quality problems to specific watershed locations may be able to change this perception. At the very least, such a model would substantiate the need for the fee and the method by which the charges were distributed.

A full survey of storm sewer infrastructure might be an overwhelming task to the operator of a small MS4, but this paper demonstrates how leveraging Python in GIS can help rapidly synthesize raw geographic data from different sources into a stormwater geodatabase. A fair rebuttal to this

argument, is that an MS4 that does not have the resources to perform a field survey of their infrastructure will probably not have the resources to write (or even use) programming code in their daily operations. However the benefit of using Python is that the code can be hidden behind a user interface that is identical to any other tool, so that an understanding of how to use GIS is all a user needs to properly run the script. The Python language is extremely readable by design, so simple syntax and inline documentation will provide the user with the necessary competency concerning the mathematical methods, while an appropriate level of constraints prevent erroneous layer input. The results of these scripts should be presented in a fashion that enables simple interpretation and visualization. For example, the Curve Number script writes the calculated CN to a new field in the Subbasin feature class so that new results might be compared to old. This also allows the symbolization of a map based on CN, providing a means for visual inspection of results. A Python script should also provide status messages in the GIS environment during computation, to assure that the process is running correctly, inform the user of any errors, and impart context on the algorithm performed. Simplicity of use, transparency of operation, and logical output should encourage adoption, and may also initiate further interest in process automation.

The authors' experience with Python programming in GIS was briefly frustrating, but the benefits were almost immediate, as repetitive tasks were performed using a 'for' loop. Once the initial frustrations were overcome, the overwhelming capabilities of automating geoprocesses promoted a paradigm of process-centered database creation. This is a clear improvement over the previous paradigm of related but disjoint process implementation. The advantage of workflow automation is clear at the process level, but even more compelling at the project level, as all small MS4s share the same objectives. A different operator could administer the workflow to their own urban area with only minimal modification based on available input data, shortening the time between project initiation and results.

Further simplification in this workflow could be realized if the mathematical model were built directly into the GIS. This strategy was employed in Whiteaker et al. (2006), with ESRI's ModelBuilder as the processing engine. This successful integration of a hydrologic model with GIS resulted in the observations that a processor must be able to operate at the feature level, and that data storage is a potential obstacle for embedded modeling. Python has the ability to read and edit at the feature level as exemplified in the Pipe Stub script (Figure 6), and can perform cumbersome array math through the numerical extension (known as "NumPy") without excessive data

reproduction. It may therefore be a practical next step to employ Python as a processor for hydrologic calculations within the GIS interface. The advantage of this for municipal operations is that the entire workflow from data collection to model results would be housed in a single software application without any need to export data to external modeling software, further improving workflow. The disadvantage is that the robust visualization and interface features of the external model would be lost, though this might be an acceptable compromise for the operator.

An advantage to condensing data management and hydrologic modeling into a single environment is that the procedural distance between database edits and model output is much shorter. The operator will therefore be able to move through the calibration process more rapidly because data will not need to be transferred between the GIS and the external modeling software every time a change is made to the database. This could provide the necessary circumstances for an automated calibration routine, which was not part of this project but would have been a vast improvement over the traditional scheme of modifying the database, updating the model, running the model, analyzing results, repeat. If this procedure was automated along with the rest of the workflow, model calibration could be reduced from a laborious, time consuming task to another Python script.

From among the many sources of uncertainty involved in model evaluation, only measurement uncertainty is addressed in this project. Model uncertainty and sensitivity are ignored because this paper does not suggest any particular model, but only the preparation leading up to the model, and the evaluation of the performance of the model after it is created (whatever variety it may be). Measurement uncertainty as it pertains to the ADV is a readily transposable idea between municipalities, whereas model uncertainty is not. Quantifying measurement uncertainty for an ADV is also much more straightforward than evaluating the profuse sources of uncertainty in a model, and may be the only source of error that is within the faculties of a small MS4 program. Modifying a model to simulate a probable error range makes the job of calibration slightly easier, and although it conflicts with the conservative culture of hydrology, Harmel et al. (2010) and Harmel and Smith (2007) assuage these concerns through the development of a model evaluation matrix. The uncertainty in a method used for stormwater management will define the extents of the decisions that can be made from the method, so it is important to acknowledge that these uncertainties exist, and account for them in as many ways as possible.

Pitt and Clark (2008) write that “The implementation of mathematical optimization will make decisions more objective and efficient, but there is still much ‘art’ remaining in successful storm-

water management." The ideas presented in this paper might reduce the amount of "art" that is needed from an MS4 operator, but it is probably more appropriate to relate the workflow and scripts to an artist's toolbox; intended to make the task easier, and improve the final product. Through consistently improving tools and methods, managers of MS4s should be able comply with EPA regulations and manage the impact of non-point source pollution more effectively. Although it is unlikely that the so-called "art" component of stormwater management is completely abated in the near future, dissemination of a simple to use, well defined, and widely applicable procedure (i.e. best practice) for stormwater management through modeling small MS4s may have national consequences for compliance, and by extension, water quality.

REFERENCES

Abdella, Y., and Alfredsen, K. (2010). A GIS toolset for automated processing and analysis of radar precipitation data. *Computers & Geosciences*, *36*(4), 422–429. doi:10.1016/j.cageo.2009.08.008.

Amaguchi, H., Kawamura, A., Olsson, J., and Takasaki, T. (2012). Development and testing of a distributed urban storm runoff event model with a vector-based catchment delineation. *Journal of Hydrology*, *420*, 205–215. doi:10.1016/j.jhydrol.2011.12.003.

Baker, M. E., Weller, D. E., and Jordan, T. E. (2006). Comparison of Automated Watershed Delineations : Effects on Land Cover Areas , Percentages , and Relationships to Nutrient Discharge. *Photogrammetric Engineering and Remote Sensing*, *72*(2), 159–168.

Ball, J. E. (2009). Discussion of "Automatic Calibration of the US EPA SWMM Model for a Large Urban Catchment" by J. Barco, K.M. Wong, and M.K. Stenstrom. *Journal of Hydraulic Engineering*, *135*(12), 1108–1110.

Bentley Systems Inc. (2013). SewerGEMs v8i [Computer Software]. Exton, PA.

Beven, K. (2008). On doing better hydrological science. *Hydrological Processes*, *22*(17), 3549–3553. doi:10.1002/hyp.

Bingeman, A., Kouwen, N., and Soulis, E. (2006). Validation of the hydrological processes in a hydrological model. *Journal of Hydrologic Engineering*, *11*(5), 451–463.

Bonakdari, H., and Zinatizadeh, A. A. (2011). Influence of position and type of Doppler flow meters on flow-rate measurement in sewers using computational fluid dynamic. *Flow Measurement and Instrumentation*, *22*(3), 225–234. doi:10.1016/j.flowmeasinst.2011.03.001.

Borah, D. K., and Bera, M. (2003). Watershed-Scale Hydrologic and Nonpoint-Source Pollution Models: Review of Mathematical Bases. *Transactions of the ASAE*, *46*(6), 1553–1566.

Borah, D. K., and Bera, M. (2004). Watershed-Scale Hydrologic and Nonpoint-Source Pollution Models: Review of Applications. *Transactions of the ASAE*, *47*(3), 789–804.

Bryan, B. A. (2013). High-performance computing tools for the integrated assessment and modelling of social–ecological systems. *Environmental Modelling & Software*, *39*, 295–303. doi:10.1016/j.envsoft.2012.02.006.

Burns, M. J., Fletcher, T. D., Walsh, C. J., Ladson, A. R., and Hatt, B. E. (2012). Hydrologic shortcomings of conventional urban stormwater management and opportunities for reform. *Landscape and Urban Planning*, *105*(3), 230–240. doi:10.1016/j.landurbplan.2011.12.012.

Cave, K. A., Bryson, D. S., and Bails, J. D. (1999). Will the New Federal Phase 2 Storm Water Program Work?: Test Case with Michigan's Voluntary General Storm Water Permit. In *WEFTEC*.

Choi, Y., Yi, H., and Park, H.-D. (2011). A new algorithm for grid-based hydrologic analysis by incorporating stormwater infrastructure. *Computers & Geosciences*, *37*(8), 1035–1044. doi:10.1016/j.cageo.2010.07.008.

Chow, V. Te, Maidment, D. R., and Mays, L. W. (1988). *Applied Hydrology* (p. 11). McGraw-Hill.

Clark, M. J. (1998). Putting water in its place : a perspective on GIS in hydrology and water management. *Hydrological Processes*, *12*(6), 823–834.

Clean Water Act. , Pub. L. No. 1251 et seq. (1972). 33 U.S.C. Retrieved from http://www.epw.senate.gov/water.pdf.

CWP. (2008). *Managing Stormwater in Your Community - A Guide for Building an Effective Post-Construction Program*.

DeVantier, B. A., and Feldman, A. D. (1993). Review of GIS applications in hydrologic modeling. *Journal of Water Resources Planning and Management*, *119*(2), 246–261.

Dewals, B., Archambeau, P., Khuat Duy, B., Erpicum, S., and Pirotton, M. (2012). Semi-explicit modelling of watersheds with urban drainage systems. *Engineering Applications of Computational Fluid Mechanics*, *6*(1), 46–57.

Djokic, D., and Maidment, D. R. (1991). Terrain Analysis for Urban Stormwater Modelling. *Hydrological Processes*, *5*(October 1990), 115–124.

Ellard, M. W. (2010). Equitable Credits for Stormwater Fee Assessment. In *Innovations in Watershed Management under Land Use and Climate Change. Proceedings of the 2010 Watershed Management Conference* (pp. 598–609). Madison, Wisconsin, USA: American Society of Civil Engineers.

Elliot, A. H., and Trowsdale, S. A. (2007). A review of models for low impact urban stormwater drainage. *Environmental Modelling & Software*, *22*(3), 394–405. doi:10.1016/j.envsoft.2005.12.005.

Emerson, C. H., Welty, C., and Traver, R. G. (2005). Watershed-Scale Evaluation of a System of Storm Water Detention Basins. *Journal of Hydrologic Engineering*, *10*(3), 237–242.

ESRI. (2008). ArcPad 7.1.1 [Computer Software]. Redlands, CA.: Environmental Systems Research Institute.

ESRI. (2013). ArcGIS 10.1 [Computer Software]. Redlands, CA.: Environmental Systems Research Institute.

Fletcher, T. D., Andrieu, H., and Hamel, P. (2012). Understanding, management and modelling of urban hydrology and its consequences for receiving waters: A state of the art. *Advances in Water Resources*. doi:10.1016/j.advwatres.2012.09.001.

Garbrecht, J., Ogden, F. L., DeBarry, P. A., and Maidment, D. R. (2001). GIS and distributed watershed models. I: Data coverages and sources. *Journal of Hydrologic Engineering*, *6*(6), 506–514.

Gironás, J., Niemann, J. D., Roesner, L. A., Rodriguez, F., and Andrieu, H. (2009). Evaluation of Methods for Representing Urban Terrain in Storm-Water Modeling. *Journal of Hydrologic Engineering*, *15*(1), 1–14.

Goring, D. G., and Nikora, V. I. (2002). Despiking Acoustic Doppler Velocimeter Data. *Journal of Hydraulic Engineering*, *128*(1), 117–126.

Gupta, H. V., Sorooshian, S., and Yapo, P. O. (1998). Toward improved calibration of hydrologic models: Multiple and noncommensurable measures of information. *Water Resources Research*, *34*(4), 751. doi:10.1029/97WR03495.

Haddock, G., and Jankowski, P. (1993). Integrating nonpoint source pollution modeling with a geographic information system. *Computers, Environment and Urban Systems*, *17*(5), 437–451. doi:10.1016/0198-9715(93)90039-8.

Halfawy, M. R. (2008). Integration of municipal infrastructure asset management processes: challenges and solutions. *Journal of Computing in Civil Engineering*, *22*(3), 216–229.

Harmel, R. D., Cooper, R. J., Slade, R. M., Haney, R. L., and Arnold, J. G. (2006). Cumulative uncertainty in measured streamflow and water quality data for small watersheds. *Transactions of the ASABE*, *49*(3), 689–701.

Harmel, R. D., Smith, D. R., King, K. W., and Slade, R. M. (2009). Estimating storm discharge and water quality data uncertainty: A software tool for monitoring and modeling applications. *Environmental Modelling & Software*, *24*(7), 832–842. doi:10.1016/j.envsoft.2008.12.006.

Harmel, R. D., and Smith, P. K. (2007). Consideration of measurement uncertainty in the evaluation of goodness-of-fit in hydrologic and water quality modeling. *Journal of Hydrology*, *337*(3-4), 326–336. doi:10.1016/j.jhydrol.2007.01.043.

Harmel, R. D., Smith, P. K., and Migliaccio, K. W. (2010). Modifying Goodness-of-Fit Indicators to Incorporate Both Measurement and Model Uncertainty in Model Calibration and Validation. *Transactions of the ASABE*, *53*(1), 55–63.

Hodgson, M. E., and Bresnahan, P. (2004). Accuracy of airborne lidar-derived elevation: empirical assessment and error budget. *Photogrammetric Engineering and Remote Sensing*, *70*(3), 331–339.

Huber, W. C., and Strecker, E. W. (2008). Urban Stormwater Modeling: Status in 2008. In *World Environmental and Water Resources Congress* (pp. 1–10). sAhupua'A: ASCE.

ISCO. (2011). 2150 Area Velocity Flow Module and Sensor - Installation and Operation Guide. Teledyne ISCO.

James, M. B., and Dymond, R. L. (2012). Bioretention Hydrologic Performance in an Urban Stormwater Network. *Journal of Hydrologic Engineering*, *17*(3), 431–436. doi:10.1061/(ASCE)HE.1943-5584.0000448.

Jin, M., Coran, S., and Cook, J. (2002). New one-dimensional implicit numerical dynamic sewer and storm model. In *Proc. 9th Int. Conf. Urban Drainage–Global Solutions for Urban Drainage* (pp. 1–9). ASCE.

Johnson, L. E. (2009). *Geographic Information Systems in Water Resources Engineering* (p. 60). Boca Raton, Florida: CRC Press.

Jones, N. L., Wright, S. G., and Maidment, D. R. (1990). Watershed delineation with triangle-based terrain models. *Journal of Hydraulic Engineering*, *116*(10), 1232–1251.

Karssenberg, D. (2002). The value of environmental modelling languages for building distributed hydrological models. *Hydrological Processes*, *16*(14), 2751–2766. doi:10.1002/hyp.1068.

Kirchner, J. W. (2006). Getting the right answers for the right reasons: Linking measurements, analyses, and models to advance the science of hydrology. *Water Resources Research*, *42*(3). doi:10.1029/2005WR004362.

Klemeš, V. (1986a). Dilettantism in hydrology: Transition or destiny? *Water Resources Research*, *22*(9S), 177S–188S.

Klemeš, V. (1986b). Operational testing of hydrological simulation models. *Hydrological Sciences Journal*, *31*(1), 13–24.

Larrarte, F., Bernard Bardiaux, J., Battaglia, P., and Joannis, C. (2008). Acoustic Doppler flow-meters: A proposal to characterize their technical parameters. *Flow Measurement and Instrumentation*, *19*(5), 261–267. doi:10.1016/j.flowmeasinst.2008.01.001.

Leopold, L. B. (1968). *Hydrology for Urban Land Planning - A Guidebook on the Hydrologic Effects of Urban Land Use* (pp. 1–18). United States. Department of the Interior.

Li, J., and Wong, D. W. (2010). Effects of DEM sources on hydrologic applications. *Computers, Environment and Urban Systems*, *34*(3), 251–261. doi:10.1016/j.compenvurbsys.2009.11.002.

Lutz, M. (1996). *Programming Python* (1st ed.). O'Reilly.

Maidment, D. R. (1996). GIS and hydrologic modeling-an assessment of progress. In *Proceedings of the Third International Conference on Integrating GIS and Environmental Modelling*. Santa Fe, NM: National Center for Geographic Information and Analysis.

Maidment, D. R. (2002). *Arc Hydro: GIS for Water Resources* (p. 203). Redlands, CA.: ESRI Press.

Martin, P. H., LeBoeuf, E. J., Dobbins, J. P., Daniel, E. B., and Abkowitz, M. D. (2005). Interfacing GIS with Water Resource Models: A State of the Art Review. *Journal of the American Water Resources Association*, *41*(6), 1471–1487.

McCuen, R. H., and Moglen, G. E. (1988). Multicriterion Stormwater Management Methods. *Journal of Water Resources Planning and Management*, *114*(4), 414–431.

Miller, S., Semmens, D., Goodrich, D., Hernandez, M., Miller, R., Kepner, W., and Guertin, D. (2007). The Automated Geospatial Watershed Assessment tool. *Environmental Modelling & Software*, *22*(3), 365–377. doi:10.1016/j.envsoft.2005.12.004.

Moglen, G. E., and Hartman, G. L. (2001). Resolution Effects on Hydrologic Modeling Parameters and Peak Discharge. *Journal of Hydrologic Engineering*, *6*(6), 490–497.

Moriasi, D. N., Arnold, J. G., Liew, M. W. Van, Bingner, R. L., Harmel, R. D., and Veith, T. L. (2007). Model Evaluation Guidelines for Systematic Quanitification of Accuracy in Watershed Simulations. *Transactions of the ASABE*, *50*(3), 885–900.

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

Ogden, F. L., Garbrecht, J., DeBarry, P. A., and Johnson, L. E. (2001). GIS and distributed watershed models. II: Modules, interfaces, and models. *Journal of Hydrologic Engineering*, *6*(6), 515–523.

Olsson, J., Amaguchi, H., Alsterhag, E., Dåverhög, M., Adrian, P.-E., and Kawamura, A. (2012). Adaptation to climate change impacts on urban storm water: a case study in Arvika, Sweden. *Climatic Change*, *116*(2), 231–247. doi:10.1007/s10584-012-0480-y.

Pitt, R., and Clark, S. E. (2008). Integrated storm-water management for watershed sustainability. *Journal of Irrigation and Drainage Engineering*, *134*(5), 548–555.

Pitt, R., Lilburn, M., Durrans, S. R., Burian, S., Nix, S., Voorhees, J., Martinson, J., and Fan, C.-Y. (1999). *Guidance Manual for Integrated Wet Weather Flow (WWF) Collection and Treatment Systems for Newly Urbanized Areas (New WWF Systems)* (p. 657). Washington D.C.

Rehmel, M. (2008). Application of Acoustic Doppler Velocimeters for Streamflow Measurements. *Journal of Hydraulic Engineering*, *133*(12), 1433–1438.

Rodriguez, F., Andrieu, H., and Morena, F. (2008). A distributed hydrological model for urbanized areas – Model development and application to case studies. *Journal of Hydrology*, *351*(3-4), 268–287. doi:10.1016/j.jhydrol.2007.12.007.

Schueler, T. R. (1987). *Controlling Urban Runoff: A Practical Manual for Planning and Designing Urban BMPs* (p. 275). Washington D.C.

SCS. (1972). Section 4: Hydrology. In *National Engineering Handbook*. Washington D.C.: USDA Soil Conservation Service.

Seth, I. (2006). Use of GIS in urban storm-water modeling. *Journal of Environmental Engineering*, (December), 1550–1553.

Shamsi, U. M. (1996). Storm-water management implementation through modeling and GIS. *Journal of Water Resources Planning and Management*, *122*(2), 114–127.

Singh, V. P., and Woolhiser, D. A. (2002). Mathematical modeling of watershed hydrology. *Journal of hydrologic engineering*, *7*(4), 270–292.

Sui, D. Z., and Maggio, R. C. (1999). Integrating GIS with hydrological modeling: practices, problems, and prospects. *Computers, Environment and Urban Systems*, *23*(1), 33–51. doi:10.1016/S0198-9715(98)00052-0.

USEPA. National Pollutant Discharge Elimination System - Regulations for Revision of the Water Pollution Control Program Addressing Storm Water Discharges (1999). Environmental Protection Agency.

USEPA. (2002). Section 319 Success Stories Volume III: The Successful Implementation of the Clean Water Act's Section 319 Nonpoint Source Pollution Program.

USEPA. (2010). MS4 Permit Improvement Guide. U.S. EPA Office of Water.

Van Der Knijff, J. M., Younis, J., and De Roo, a. P. J. (2010). LISFLOOD: a GIS-based distributed model for river basin scale water balance and flood simulation. *International Journal of Geographical Information Science*, *24*(2), 189–212. doi:10.1080/13658810802549154.

Vanier, D. "Dana". (2001). Why industry needs asset management tools. *Journal of Computing in Civil Engineering*, *15*(1), 35–43.

VanRossum, G. (2013). Python programming language. The Python Software Foundation. Retrieved from http://www.python.org/ (accessed 3/18/2013).

Vermeyen, T. B. (2004). A Laboratory Evaluation of Unidata's Starflow Doppler Flowmeter and MGD Technologies' Acoustic Doppler Flow Meter. In *Water Resources 2000 - Building Partnerships* (pp. 1–10).

Wang, L., and Yu, J. (2012). Modelling detention basins measured from high-resolution light detection and ranging data. *Hydrological Processes*, *26*(19), 2973–2984. doi:10.1002/hyp.8314.

White, C. M. (2011). *Analysis and Comparison of a Detailed Land Cover Dataset versus the National Land Cover Dataset (NLCD) in Blacksburg , Virginia*. Virginia Polytechnic and State University.

White, S. S., and Boswell, M. R. (2006). Planning for Water Quality: Implementation of the NPDES Phase II Stormwater Program in California and Kansas. *Journal of Environmental Planning and Management*, *49*(1), 141–160. doi:10.1080/09640560500373386.

Whiteaker, T. L., Maidment, D. R., Goodall, J. L., and Takamatsu, M. (2006). Integrating Arc Hydro Features with a Schematic Network. *Transactions in GIS*, *10*(2), 219–237. doi:10.1111/j.1467-9671.2006.00254.x.

Yang, C. T. (1996). *Sediment Transport: Theory and practice*. New York: McGraw-Hill.

APPENDICES

Appendices A-E are the complete code blocks for the scripts mentioned in this thesis. The Python scripts alone are useful for understanding workflow, but in order to implement the scripts, they should be packaged in an ArcToolbox so that a suitable level of constraint is imposed on the layer parameterization occurring in the GIS environment. The scripts as shown use the actual names of geoprocesses, but the input layers and field names are as generic as possible.

Appendix F includes goodness-of-fit calculations for the three storm events presented in Section 3.3 for calibration of the H&H model.

APPENDIX A – CURVE NUMBER SCRIPT

```
import arcpy
arcpy.env.workspace = arcpy.GetParameterAsText(0) #Input Destination for all the junk
      FCs (Scratch GDB)
arcpy.env.overwriteOutput = True

Subbasins = arcpy.GetParameterAsText(1) #Input Subbasin Feature Layer
Soils = SSURGO.shp #static location for soils features
LandCover = arcpy.GetParameterAsText(2) #Input Land Cover Polygon Feature Layer
RelationTable = RelationTable.csv #Static Location for Lookup table for HSG-LC
      combination.  Must be established by user

arcpy.AddField_management(Subbasins,"Orig_Area","DOUBLE")
arcpy.CalculateField_management(Subbasins,"Orig_Area","!Shape_Area!","PYTHON")

#Intersect Subbasins, Land Cover, and Soils, then Create the LC-HSG field - THE CALC
      FIELD IS SENSITIVE TO THE INTERSECT NAMING CONVENTION
arcpy.Intersect_analysis([Soils,LandCover,Subbasins],"Intersect","ALL")
arcpy.AddField_management("Intersect","LC_HSG","TEXT")
arcpy.CalculateField_management("Intersect","LC_HSG",'[AREA_CODE]&"-
      "&[HSG_modifi]',"VB")

#Create a Feature Layer of the Intersect FC and join to lookup table to attribute CNs
arcpy.AddField_management("Intersect","CN_New","DOUBLE")
arcpy.MakeFeatureLayer_management("Intersect","IntersectLayer")
arcpy.AddJoin_management("IntersectLayer","LC_HSG",RelationTable,"LCHSG")
arcpy.CalculateField_management("IntersectLayer","Intersect.CN_New","[DetailedCNRelat
      ionTable.csv.CN]","VB")

#Calculate fractional area of original shape and area weighted CN
arcpy.AddField_management("Intersect","AreaRatio","DOUBLE")
arcpy.CalculateField_management("Intersect","AreaRatio","[Shape_Area]/[Orig_Area]","V
      B")
arcpy.AddField_management("Intersect","CNWeight","DOUBLE")
arcpy.CalculateField_management("Intersect","CNWeight","[CN_New]*[AreaRatio]","VB")
#Dissolve based on Basin ID to re-form original polygons with CNs.  Add statistics to
check unity
arcpy.Dissolve_management("Intersect","Dissolve","BasinID",[["AreaRatio","SUM"],["CNW
      eight","SUM"],["Shape_Area","SUM"],["CN_New","STD"]])

#Joins the Dissolve FC to the Original Subbasins FC and writes the CN to a field
called NewCN to be copied as desired
arcpy.AddField_management(Subbasins,"NewCN","DOUBLE")
arcpy.AddField_management(Subbasins,"CNStdDev","DOUBLE")
arcpy.MakeFeatureLayer_management(Subbasins,"SubbasinsLayer")
arcpy.MakeFeatureLayer_management("Dissolve","DissolveLayer")
arcpy.AddJoin_management("SubbasinsLayer","BasinID","DissolveLayer","BasinID","KEEP_A
      LL")
arcpy.CalculateField_management("SubbasinsLayer","NewCN","[Dissolve.SUM_CNWeight]","V
      B")
arcpy.CalculateField_management("SubbasinsLayer","CNStdDev","[Dissolve.STD_CN_New]","
      VB")
arcpy.DeleteField_management(Subbasins,"Orig_Area")
```

APPENDIX B – STAGE - STORAGE CURVE SCRIPT

```
import arcpy
import csv

arcpy.env.workspace = arcpy.GetParameterAsText(0) #Input Destination for all the
      junk FCs (Scratch GDB)
arcpy.env.overwriteOutput = True

Surface = arcpy.GetParameterAsText(1) #Input TIN containing elevation information
      for selected area
ElevInt = arcpy.GetParameterAsText(2) #Integer value for elevation interval
ContourName = arcpy.GetParameterAsText(3) #Name for output Contour FC
outfile = " Stage_Storage.csv" #location for output csv file

arcpy.SurfaceContour_3d(Surface, ContourName,ElevInt) #create polyline contours of
      the surface
arcpy.Dissolve_management(ContourName,ContourName+"_Dissolve","Contour")   #dissolve
      based on contour field to create a list of elevations

#Create Elevation, Area, Volume table as GIS Output and Python List
list = [] #create blank list
rows = arcpy.gp.searchCursor(ContourName+"_Dissolve","","","Contour")
arcpy.AddMessage("Elevation,2D Area (sq. ft.),Volume (cu. ft.)")
list.append(("Elevation","2D Area (sq. ft.)","Volume (cu. ft.)"))
for row in rows: #Repeat process and write to list
    result1 = arcpy.SurfaceVolume_3d(Surface,"","BELOW",int(row.Contour))
    text = result1.getMessage(2)
    area = text[text.find("2D Area")+8:text.find("2D Area")+17]
    volume = text[text.find("Volume=")+7:text.find("Volume=")+17]
    arcpy.AddMessage(str(row.Contour)+","+area+","+volume)
    list.append((str(row.Contour), area, volume))
#Write Python list to csv file for external use
with open(outfile, 'wb') as csvoutput:
    writer = csv.writer(csvoutput, delimiter = ",", lineterminator = '\n')
    writer.writerows(list)
csvoutput.close()
```

APPENDIX C – TIME OF CONCENTRATION SCRIPT

```
import arcpy
arcpy.env.workspace = arcpy.GetParameterAsText(0) #Input Destination for all the
      junk FCs (Scratch GDB)
arcpy.env.overwriteOutput = True
workfolder = arcpy.env.workspace

Flowpaths2D = arcpy.GetParameterAsText(1) #Input Original 2D flowpath FC with Sheet
      flow separated
Subbasins = arcpy.GetParameterAsText(2) #Input Subbasin Feature Layer
Pipes = arcpy.GetParameterAsText(3) #Input Pipes Feature Layer
LandCover = arcpy.GetParameterAsText(4) #Input Land Cover Polygon Feature Layer
Surface = arcpy.GetParameterAsText(5) #Input TIN
spatialRef = arcpy.Describe(Flowpaths2D).spatialReference

#Create a Feature Class to write to later using the correct field format
arcpy.CreateFeatureclass_management(workfolder,"Tc_00_Format","POLYLINE")
Flowpaths = "Tc_00_Format"
arcpy.AddField_management(Flowpaths,"FlowType","TEXT")
arcpy.AddField_management(Flowpaths,"LandCover","TEXT")
arcpy.AddField_management(Flowpaths,"BasinID","SHORT")
arcpy.AddField_management(Flowpaths,"Alternative","SHORT")

#The user draws several potential flowpaths in Flowpaths2D for each subbasin.  #The
paths must be divided into three flow regimes:
#   1. Sheet
#   2. Shallow
#   3. Channel
#Sheet flow should be no longer than 100 meters (VDOT Standard).  Shallow #connects
the end of sheet flow with the most upstream inlet.  Channel flow #should have
geometry coincident to pipes and channels, i.e. vertices should be #snapped to any
node along the channel.  This is very important...the intersect #will not work if
the geometry of these flow paths is not exactly coincident to #the conveyances.

#Intersect of Flowpaths and Subbasins so each feature gets a BasinID
arcpy.Intersect_analysis([Flowpaths2D,Subbasins],"Tc_01_Intersect","ALL")

#Identity with Pipe feature class - this assigns Pipe attributes to the #Flowpaths
if there is an intersect, otherwise it leaves the pipe attribute #fields blank.
arcpy.Identity_analysis("Tc_01_Intersect",Pipes,"Tc_02_Identity","ALL")

#Must create a feature layer for selection purposes
arcpy.MakeFeatureLayer_management("Tc_02_Identity","IdentityLayer")

#Select Features that intersect pipes, and write "Pipe" to LandCover field
arcpy.SelectLayerByAttribute_management("IdentityLayer","NEW_SELECTION","""  "Type" =
      'Closed Conduit' """)
arcpy.CalculateField_management("IdentityLayer","LandCover",' "Pipe" ')

#Select Features that intersect channels, and write "Grass" to LandCover field
arcpy.SelectLayerByAttribute_management("IdentityLayer","NEW_SELECTION","""  "Type" =
      'Open Channel' """)
arcpy.CalculateField_management("IdentityLayer","LandCover",' "Grass" ')
```

```
#Select Channel Flowpaths and Append to Flowpath FC
arcpy.SelectLayerByAttribute_management("IdentityLayer","NEW_SELECTION","""
      "FlowType" = 'Channel' """)
arcpy.Append_management("IdentityLayer",Flowpaths,"NO_TEST","")

#Select Sheet and Shallow Flowpaths and intersect with Land Cover, Reclass into
      LandCover field
arcpy.SelectLayerByAttribute_management("IdentityLayer","NEW_SELECTION","""
      "FlowType" = 'Sheet' OR "FlowType" = 'Shallow' """)
arcpy.Intersect_analysis(["IdentityLayer",LandCover],"Tc_03_Intersect","ALL")

expression = "Reclass(!AREATYPE!)"
codeblock = """def Reclass(AreaType):
  if (AreaType <= 5):
    return "Paved"
  elif (AreaType >= 6):
    return "Unpaved" """
arcpy.CalculateField_management("Tc_03_Intersect","LandCover", expression, "PYTHON",
      codeblock)

#Append Sheet and Shallow Flowpaths to Flowpath FC
arcpy.Append_management("Tc_03_Intersect",Flowpaths,"NO_TEST","")
arcpy.AddMessage("Flowpaths FC Complete")

#Convert to 3D polyline FC, add surface information
arcpy.InterpolateShape_3d(Surface, Flowpaths, "Tc_04_Interpolate")
arcpy.AddSurfaceInformation_3d("Tc_04_Interpolate",                          Surface,
      ["Z_MIN","Z_MAX","SURFACE_LENGTH"],"LINEAR")
arcpy.AddMessage("Flowpaths3D Created, Surface Information Added")

#Calculate Travel Time
arcpy.AddField_management("Tc_04_Interpolate","TcMin","DOUBLE")
arcpy.AddField_management("Tc_04_Interpolate","Slope","DOUBLE")
arcpy.CalculateField_management("Tc_04_Interpolate","Slope","(!Z_Max!-
      !Z_Min!)/!SLength!","PYTHON")
```

APPENDIX D – PIPE STUB SCRIPT

```
import arcpy
import math
import datetime

#Format today's date for concatentation to output filename
date = str(datetime.date.today())
date = date.replace("-","")

#Set up workspaces
arcpy.env.workspace  =  arcpy.GetParameterAsText(0)  #Input  Destination  for  the
      PipeStub FC
arcpy.env.overwriteOutput = True

SWStructures = arcpy.GetParameterAsText(1) #Input SWStructures FC
StubLength = arcpy.GetParameterAsText(2) #Input Stub Length
StubLength = int(StubLength)
outFC = "PipeStub"+date+".shp"
spatialRef = arcpy.Describe(SWStructures).spatialReference

#create new lineFC and insert cursor
arcpy.CreateFeatureclass_management(arcpy.env.workspace,outFC,
      "POLYLINE","","DISABLED","DISABLED",spatialRef)
arcpy.AddField_management (outFC, "ID" , "LONG", "", "", 6)
arcpy.AddField_management (outFC, "CONVTYPE" , "TEXT", "", "", 50)
arcpy.AddField_management (outFC, "NUM_CONV" , "TEXT", "", "", 50)
arcpy.AddField_management (outFC, "CONVSHAPE" , "TEXT", "", "", 50)
arcpy.AddField_management (outFC, "CONVMAT" , "TEXT", "", "", 50)
arcpy.AddField_management (outFC, "CONVSIZE" , "LONG", "", "", 6)
arcpy.AddField_management (outFC, "CONVSIZE2" , "LONG", "", "", 6)
arcpy.AddField_management (outFC, "LENGTHFT" , "DOUBLE")
arcpy.AddField_management (outFC, "INVDEPTH" , "DOUBLE")
arcpy.AddField_management (outFC, "COMMENTS" , "TEXT", "", "", 200)

#Generates Insert Cursor on Pipe Stub feature class
InsCur = arcpy.InsertCursor(outFC)
lineArray = arcpy.Array()

#insert SearchCursor in Structures FC
rows = arcpy.SearchCursor(SWStructures)

#Identify the geometry field in Structures
desc = arcpy.Describe(SWStructures)
shapefieldname = desc.ShapeFieldName
fields = desc.fields

for row in rows: #Iterate throught the rows (features) in the SWStructures FC
    feat = row.getValue(shapefieldname)
    pnt = feat.getPart() #access point geometry
    numConv = 0 #Initialize numConv variable

    for field in fields: #Iterate through the fields in SWStructures
        if field.name == "NUMCONV":
            numConv = row.getValue(field.name)
```

```
        for i in range (1, numConv+1): #Iterate through each conveyance, assigning
            attribute values
            if field.name == "C"+str(i)+"AZIMUTH":
                azimuth = row.getValue(field.name)
                #assign variables for the line start point
                x1, y1 = [pnt.X, pnt.Y]
                #assign variables for the line stop point
                x2 = x1+StubLength*math.sin(math.radians(azimuth))
                y2 = y1+StubLength*math.cos(math.radians(azimuth))
                #start point in lineArray
                start = arcpy.Point()
                (start.ID, start.X, start.Y) = (1, x1, y1)
                lineArray.add(start)
                #end point in lineArray
                end = arcpy.Point()
                (end.ID, end.X, end.Y) = (2, x2, y2)
                lineArray.add(end)
                 #write feature to shapefile
                feature = InsCur.newRow()
                feature.shape = lineArray
                #clear lineArray
                lineArray.removeAll()
            #Assign the rest of the attributes for individual conveyances to Pipe FC
            if field.name == "C"+str(i)+"TYPE":
                type = row.getValue(field.name)
                feature.CONVTYPE = type
            if field.name == "C"+str(i)+"NUM_CONV":
                numconv = row.getValue(field.name)
                feature.NUM_CONV = numconv
            if field.name == "C"+str(i)+"MATERIAL":
                material = row.getValue(field.name)
                feature.CONVMAT = material
            if field.name == "C"+str(i)+"DIM1":
                dim1 = row.getValue(field.name)
                feature.CONVSIZE = dim1
            if field.name == "C"+str(i)+"DIM2":
                dim2 = row.getValue(field.name)
                feature.CONVSIZE2 = dim2
            if field.name == "C"+str(i)+"SHAPE":
                shape = row.getValue(field.name)
                feature.CONVSHAPE = shape
            if field.name == "C"+str(i)+"DEPTH":
                depth = row.getValue(field.name)
                feature.INVDEPTH = depth
                InsCur.insertRow(feature)
del InsCur
del rows
```

APPENDIX E – STATION – ELEVATION CURVE SCRIPT

```
import arcpy
import math
import os
import csv

workfolder = arcpy.GetParameterAsText(0) #Input Destination for working feature classes
arcpy.env.workspace = workfolder
arcpy.env.overwriteOutput = True

StreamFC = arcpy.GetParameterAsText(1) #input lineFC with open channel sections selected
Nodes = arcpy.GetParameterAsText(2) #input Nodes FC to join IDs
NodeIDField = arcpy.GetParameterAsText(3) #Select field in Nodes FC that stores Unique Identifier
surface = arcpy.GetParameterAsText(4) #input TIN Surface to extract elevations
outFC = arcpy.GetParameterAsText(5) #output line feature class showing locations of cross sections
outFolder = arcpy.GetParameterAsText(6) #destination folder for cross section ASCII files
SectionLength = arcpy.GetParameterAsText(7) #Input Section Length
SectionLength = int(SectionLength)
outfilename = 'Compiled.csv'

spatialRef = arcpy.Describe(StreamFC).spatialReference

#create new lineFC and create insert cursor
arcpy.CreateFeatureclass_management(workfolder, "CrossSection2D", "Polyline", "", "DISABLED","DISABLED", spatialRef)

#This section reads the stream features and writes projection lines to the newly created feature class
InsCur = arcpy.InsertCursor("CrossSection2D")
lineArray = arcpy.Array()
desc = arcpy.Describe(StreamFC)
shapefieldname = desc.ShapeFieldName
rows = arcpy.SearchCursor(StreamFC)
for row in rows:
    # Create the geometry object
    feat = row.getValue(shapefieldname)
    partnum = 0
    # Step through each part of the feature
    for part in feat:
        #create an array that stores x, y coordinates of each vertex in the stream lines
        point_list = []
        pointNum = 0
        for pnt in feat.getPart(partnum):
            point_list.append(pnt.X)
            point_list.append(pnt.Y)
            pointNum += 1
        partnum += 1
        #Read the first two points of the array
```

```
        x_0 = point_list[0]
        y_0 = point_list[1]
        x_1 = point_list[2]
        y_1 = point_list[3]
        #Calculate x and y components
        xdisp_1 = x_1-x_0
        ydisp_1 = y_1-y_0
        if ydisp_1 == 0:
            ydisp_1 = ydisp_1 + 0.000000001 #check for horizontal line
        #Calculate azimuths of stream lines
        azimuth_1 = math.degrees(math.atan(xdisp_1/ydisp_1))
        #Add 90 degrees
        theta_1 = math.radians(azimuth_1 + 90)
        #Resolve endpoints for section line
        s_x1 = x_0 - SectionLength * math.sin(theta_1)
        s_y1 = y_0 - SectionLength * math.cos(theta_1)
        s_x2 = x_0 + SectionLength * math.sin(theta_1)
        s_y2 = y_0 + SectionLength * math.cos(theta_1)
        #start point in lineArray
        start = arcpy.Point()
        (start.ID, start.X, start.Y) = (1, s_x1, s_y1)
        lineArray.add(start)
        #end point in lineArray
        end = arcpy.Point()
        (end.ID, end.X, end.Y) = (2, s_x2, s_y2)
        lineArray.add(end)
        #write feature to shapefile
        feature = InsCur.newRow()
        feature.shape = lineArray
        #clear lineArray
        lineArray.removeAll()
        InsCur.insertRow(feature)
arcpy.AddMessage("Section feature class created")
del rows
del InsCur

#Spatial Join the section features with the relevant node
arcpy.SpatialJoin_analysis("CrossSection2D",Nodes,"CrossSectionsJoin","JOIN_ONE_TO_ON
E","KEEP_ALL")
arcpy.AddMessage("Spatial Join Complete")

#Process: Interpolate Shape - add 3D information to sections
arcpy.InterpolateShape_3d(surface, "CrossSectionsJoin", outFC, "", "1", "LINEAR",
"DENSIFY", "0")
arcpy.AddMessage("Interpolate Shape Completed")

arcpy.MakeFeatureLayer_management(outFC,"CrossSectionLayer")
#Create a search cursor on the cross sections
rows = arcpy.SearchCursor("CrossSectionLayer")
desc = arcpy.Describe("CrossSectionLayer")
fields = desc.fields

i = 1 #Create a counter
for row in rows:
```

```
    #select the row that the cursor is on currently based on OID
    arcpy.SelectLayerByAttribute_management("CrossSectionLayer","NEW_SELECTION","""
    "OBJECTID" = """+str(i))
    #Extract ID from current row
    for field in fields:
        if field.name == NodeIDField:
            ID = row.getValue(field.name)
    #Write ASCII file using ID as unique identifier
    arcpy.FeatureClassZToASCII_3d("CrossSectionLayer",                    outFolder,
    "ID_"+str(ID)+"_FID_.txt", "PROFILE", "COMMA", "AUTOMATIC", "3", "DECIMAL_POINT")
    i += 1
del rows

#Compile all ASCII files into necessary CSV format
outfile = outFolder+'\\'+outfilename
listing = os.listdir(outFolder)
writefile = open(outfile, 'wb')
write = csv.writer(writefile, delimiter = ",")
for file in listing:
    if file[0:2] == "ID":
        ASCIIfile = open(outFolder+"\\"+file,'r')
        read = csv.reader(ASCIIfile, delimiter = ",")
        split = file.split("_")
        ID = split[1]
        for row in read:
            list = []
            list.append((ID,row[0],row[1]_)
            write.writerow(list)
        ASCIIfile.close()
writefile.close()
#delete working feature classes
arcpy.Delete_management("CrossSection2D")
arcpy.Delete_management("CrossSectionsJoin")

#This section reads the output, and modifies the datum so all inverts are at 0
#elevation
import csv
outfile = "W://01-Modeling//Data//Cross_Sections_adj.csv"
joinfile = "W://01-Modeling//Data//Cross_Sections_Join.csv"
rows = []
with open(outfile, "rb") as searchfile: #open cross section text file
    filereader = csv.reader(searchfile, delimiter = ',')
    filereader.next() #skip header
    for i, line in enumerate(filereader):
        rows.append((i,int(line[0]),line[1],line[2])) #Write .txt data to #python
        list
searchfile.close() #close file
rows.append((i+1,10000,10000,10000))#add dummy row that ends the loop
end = rows[-1][0] #find index of last row in rows
list=[] #initiate a blank list that stores all the invert for each ID
dissolve_row = [] #initiate a blank list that stores ID, Invert, Ground for #each ID
IDs = []
inverts = []
grounds = []
```

```
for i in range(0,end):
    if (rows[i][1] == rows[i+1][1]): #logic to determine when to stop writing to list
        list.append(rows[i][3])
    else:
        IDs.append(rows[i-1][1])
        inverts.append(min(list))
        grounds.append(max(list))
        dissolve_row.append((rows[i-1][1],min(list),max(list)))
        list=[] #clear list for next ID
```

APPENDIX F – GOODNESS OF FIT CALCULATIONS

Calculating deviation, e_i of the *ith* measurement:

$$e_i = \begin{cases} 0 & if\ O_{i,l} \le P_i \le O_{i,u} \\ O_{i,l} - P_i & if\ P_i \le O_{i,l} \\ O_{i,u} - P_i & if\ P_i > O_{i,u} \end{cases} \quad (2)$$

The values of e_i can then be used to find the Nash-Sutcliffe correlation, R^2_{NS}:

$$R^2_{NS} = 1.0 - \frac{\sum_{i=1}^{n}(e_i)^2}{\sum_{i=1}^{n}(O_i - \bar{O})^2} \quad (3)$$

Where:

R^2_{NS} = Nash Sutcliffe Correlation coefficient (maximum = 1)

e_i = *ith* Deviation from observed discharge (cfs)

$O_{i.l}$ = Lower bound of uncertainty range for *ith* observed discharge (cfs)

$O_{i,u}$= Upper bound of uncertainty range for *ith* observed discharge (cfs)

P_i = *ith* simulated discharge (cfs)

O_i = *ith* Observed discharge (cfs)

$\bar{O}$ = Mean Observed discharge (cfs)

To quantify the volume error as a percent of total measured volume, first find the absolute deviation, e_{vol} from the observed value, similar to (2)

$$e_{vol} = \begin{cases} 0 & if\ O_{vol,l} \le P_{vol} \le O_{vol,u} \\ O_{vol,l} - P_{vol} & if\ P_{vol} \le O_{vol,l} \\ O_{vol,u} - P_{vol} & if\ P_{vol} > O_{vol,u} \end{cases} \quad (4)$$

Then use e_{vol} to calculate volume percentage error:

$$E_{vol} = \frac{e_{vol}}{O_{vol}} \times 100 \quad (5)$$

Where:

E_{vol} = Percentage volume error (%)

e_{vol} = Volume deviation from measured values (ft^3)

$O_{vol.l}$ = Lower bound of uncertainty range for volume (ft^3)

$O_{vol,u}$= Upper bound of uncertainty range for volume (ft^3)

P_{vol} = Simulated Volume (ft^3)

O_{vol} = Observed Volume (ft^3)

Finally, use the same principles to solve for peak error as a percentage of the observed peak:

$$e_{peak} = \begin{cases} 0 & if\ O_{peak,l} \leq P_{peak} \leq O_{peak,u} \\ O_{peak,l} - P_{peak} & if\ P_{peak} \leq O_{peak,l} \\ O_{peak,u} - P_{peak} & if\ P_{peak} > O_{peak,u} \end{cases} \quad (6)$$

And use e_{peak} to calculate peak percentage error:

$$E_{peak} = \frac{e_{peak}}{O_{peak}} \times 100 \quad (7)$$

Where:

E_{peak} = Percentage peak error (%)

e_{peak} = Peak deviation from measured values (cfs)

$O_{peak.l}$ = Lower bound of uncertainty range for peak (cfs)

$O_{peak,u}$= Upper bound of uncertainty range for peak (cfs)

P_{peak} = Simulated Peak (cfs)

O_{peak} = Observed Peak (cfs)