THESIS

ENHANCED WATERSHED MODELING AND DATA ANALYSIS WITH A FULLY COUPLED HYDROLOGIC MODEL AND CLOUD-BASED FLOW ANALYSIS

Submitted by

Tyler Wible

Department of Civil and Environmental Engineering

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Summer 2014

ABSTRACT

ENHANCED WATERSHED MODELING AND DATA ANALYSIS WITH A FULLY COUPLED HYDROLOGIC MODEL AND CLOUD-BASED FLOW ANALYSIS

In today’s world of increased water demand in the face of population growth and climate change, there are no simple answers. For this reason many municipalities, water resource engineers, and federal analyses turn to modeling watersheds for a better understanding of the possible outcomes of their water management actions. The physical processes that govern movement and transport of water and constituents are typically highly nonlinear. Therefore, improper characterization of a complex, integrated, processes like surface-subsurface water interaction can substantially impact water management decisions that are made based on existing models. Historically there have been numerous tools and watershed models developed to analyze watersheds or their constituent components of rainfall, run-off, irrigation, nutrients, and stream flow. However, due to the complexity of real watershed systems, many models have specialized at analyzing only a portion of watershed processes like surface flow, subsurface flow, or simply analyzing local monitoring data rather than modeling the system.

As a result many models are unable to accurately represent complex systems in which surface and subsurface processes are both important. Two popular watershed models have been used extensively to represent surface processes, SWAT (Arnold et al, 1998), and subsurface processes, MODFLOW (Harbaugh, 2005). The lack of comprehensive watershed simulation has led to a rise in uncertainty for managing water resources in complex surface-subsurface driven watersheds. For this reason, there have been multiple attempts to couple the SWAT and

MODFLOW models for a more comprehensive watershed simulation (Perkins and Sophocleous, 1999; Menking, 2003; Galbiati et al., 2006; Kim et al., 2008); however, the previous couplings are typically monthly couplings with spatial restrictions for the two models. Additionally, most of these coupled SWAT-MODFLOW models are unavailable to the general public, unlike the constituent SWAT and MODFLOW models which are available. Furthermore, many of these couplings depend on a forced equal spatial discretization for computational units. This requires that one MODFLOW grid cell is the same size and location of one SWAT hydrologic response unit (HRU). Additionally, many of the previous couplings are based on a loose monthly average coupling which might be insufficient in natural spring and irrigated agricultural driven groundwater systems which can fluctuate on a sub-monthly time scale.

The primary goal of this work is to enhance the capacity for modeling watershed processes by fully coupling surface and subsurface hydrologic processes at a daily time step. The specific objectives of this work are 1) to examine and create a general spatial linkage between SWAT and MODFLOW allowing the use of spatially-different existing models for coupling; 2) to examine existing practices and address current weaknesses for coupling of the SWAT and MODFLOW models to develop an integrated modeling system; 3) to demonstrate the capacity of the enhanced model compared to the original SWAT and MODFLOW models on the North Fork of the Sprague River in the Upper Klamath Basin in Oregon.

The resulting generalized daily coupling between a spatially dis-similar SWAT and MODFLOW model on the North Fork of the Sprague River has resulted in a slightly more lower representation of monthly stream flow (monthly $R^2 = 0.66$, NS = 0.38) than the original SWAT model (monthly $R^2 = 0.60$, NS = 0.57) with no additional calibration. The Log10 results of stream flow illustrate an even greater improvement between SWAT-MODFLOW correlation

(R^2) but not the overall simulation (NS) (monthly R^2 = 0.74, NS = -0.29) compared to the original SWAT (monthly R^2 = 0.63, NS = 0.63) correlation (R^2). With an improved water table representation, these SWAT-MODFLOW simulation results illustrate a more in depth representation of overall stream flows on a groundwater influenced tributary of the Sprague River than the original SWAT model.

Additionally, with the increased complexity of environmental models there is a need to design and implement tools that are more accessible and computationally scalable; otherwise their use will remain limited to those that developed them. In light of advancements in cloud-computing technology a better implementation of modern desktop software packages would be the use of scalable cloud-based cyberinfrastructure, or cloud-based environmental modeling services. Cloud-based deployment of water data and modeling tools assist in a scalable as well as platform independent analysis; meaning a desktop, laptop, tablet, or smart phone can perform the same analyses. To utilize recent advancements in computer technology, a further focus of this work is to develop and demonstrate a scalable cloud-computing web-tool that facilitates access and analysis of stream flow data. The specific objectives are to 1) unify the various stream flow analysis topics into a single tool; 2) to assist in the access to data and inputs for current flow analysis methods; 3) to examine the scalability benefits of a cloud-based flow analysis tool.

Furthermore, the new Comprehensive Flow Analysis tool successfully combined time-series statistics, flood analysis, base-flow separation, drought analysis, duration curve analysis, and load estimation into a single web-based tool. Preliminary and secondary scalability testing has revealed that the CFA analyses are scalable in a cloud-based cyberinfrastructure environment to a request rate that is likely unrealistic for web tools.

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Mazdak Arabi for the opportunity to have worked on this project as well has his help throughout developing the Comprehensive Flow Analysis tool and coupling of SWAT and MODFLOW. I would also like to thank Dr. Ryan Bailey and his tireless support and help on all the little problems I encountered along the way. Finally I would like to thank Jeff Ditty, without whom I don't think I would have made it this far thanks to his help, effort, and support over the years.

I really appreciated the help and the initial models and calibration by Rosie Records for SWAT on the Sprague River Basin and Marshall Gannett for MODFLOW for the Upper Klamath Basin. I would also like to thank Jeff Arnold for his support and help throughout working with/in the SWAT source code, and Andre Dozier for his general help with SWAT, MODFLOW, and FORTRAN compiling in general. I would also like to thank Dave Patterson, Kyle Traff, Wes Lloyd, and Olaf David for their help integrating the CFA tool into CSIP and eRAMS as well as general coding and testing help for the tool and interface. Additionally I would like to thank Tyler Dell for his help designing and implementing the Load Estimator (LOADEST) analysis in the CFA tool as well as Chenda (Peter) Deng's help with searching various database for common watershed models and references. I would also like to thank Rosie Records and Mehdi Ahmadi for their help with calibration statistics and general support for my coupled SWAT-MODFLOW model work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

A continuing focus of water resources planning and management is assessing the impacts of changing land use and climate conditions on stream flow. At the heart of this type of analysis is the use of existing water data and various tools to analyze things like the base-flow contribution and model these watershed systems. Many studies have focused on the impacts of land use and/or climate change on both the hydrological response of a system (Lettenmaier et al., 1999; Wood et al., 2004; Hay et al., 2011) and water quality changes within the system (Jeppsen et al., 2007; Solheim et al., 2010; Ahmadi et al., 2013). Various other topics examined are the impacts and severity of flooding (IACWD, 1982), droughts (Salas et al., 2005), and different point and non-point source stream pollution impacts (Cleland, 2007).

The physical processes that govern movement and transport of water and constituents are typically highly nonlinear resulting in complex algorithms to attempt to simulate these processes. An example of this non-linearity is the interaction between surface and subsurface water process which recursively alters soil percolation based on soil moisture content as affected by the depth to groundwater which increases with greater soil percolation. Therefore, improper characterization of these integrated processes can substantially impact water management decisions that are made based on existing models. A further hindrance to decision making is a generally poor accessibility to available flow analysis tools and insufficient infrastructure for the tool to support on-demand scalability.

Historically there have been numerous tools and watershed models developed to analyze watersheds or their constituent components of rainfall, run-off, irrigation, nutrients, and stream flow. However, due to the complexity of real watershed systems, many models have specialized at analyzing only a portion of watershed processes like surface flow, subsurface flow, or simply analyzing local monitoring data rather than modeling the system. As a result many models are unable to accurately represent complex systems in which surface and subsurface processes are both important. The lack of comprehensive watershed simulation has led to a rise in uncertainty for managing water resources in complex surface-subsurface driven watersheds. Without the inclusion of both surface and groundwater hydrologic processes, decisions based on model results may not accurately reflect actual conditions in the watershed. The best way to solve this problem would be to integrate a surface process driven model with a subsurface process driven model on a refined temporal scale for a more comprehensive watershed simulation.

Two popular watershed models have been used extensively to represent surface processes, SWAT (Arnold et al, 1998), and subsurface processes, MODFLOW (Harbaugh, 2005). The popularity and focus of these models has led to multiple attempts to couple the models (Perkins and Sophocleous, 1999; Menking, 2003; Galbiati et al., 2006; Kim et al., 2008); however, the previous couplings are typically monthly couplings with spatial restrictions for the two models. Additionally, most of these coupled SWAT-MODFLOW models are unavailable to the general public, unlike the constituent SWAT and MODFLOW models which are available. Furthermore, many of these couplings depend on a forced equal spatial discretization for computational units (Kim et al., 2008; Chung et al., 2010). This requires that 1 MODFLOW grid cell is the same size and location of 1 SWAT hydrologic response unit (HRU). Additionally, many of the previous couplings are based on a monthly average coupling which might be insufficient in natural spring and irrigated agricultural driven groundwater systems in which water table elevation can fluctuate on a daily basis.

The primary goal of this work is to enhance the capacity for modeling watershed processes by fully coupling surface and subsurface hydrologic processes at a daily time step. The specific objectives of this work are 1) to examine and create a general spatial linkage between SWAT and MODFLOW allowing the use of spatially-different existing models for coupling; 2) to examine existing practices and address current weaknesses for coupling of the SWAT and MODFLOW models to develop an integrated modeling system; 3) to demonstrate the capacity of the enhanced model compared to the original SWAT and MODFLOW models. The performance of the coupled SWAT-MODFLOW model will be tested in the Upper Klamath Basin due to the complex groundwater interactions coupled with an interest in surface hydrology problems.

Additionally, with the increased complexity of models of the environment there is a need to design and implement tools that are more accessible and computationally scalable. Otherwise their use will remain limited to those that developed them. Most models and analyses are desktop-based software packages that use local computer resources to execute. Recently there has been a shift in the implementation of these analyses to port them to the web with web-based graphical user interfaces to assist in with interacting with inputs and results of the analysis. However, in light of advancements in cloud-computing technology an even better implementation would be the use of scalable cloud-based cyberinfrastructure. Cloud-based deployment of water data and modeling tools assist in a scalable, as well as, platform independent analysis; meaning a desktop, laptop, tablet, or smart phone can perform the same analyses.

To utilize recent advancements in computer technology, the focus of this work was to develop and demonstrate a scalable cloud-computing web-tool that facilitates access and analysis of stream flow data. The specific objectives are to 1) unify the various stream flow analysis topics into a single tool; 2) to assist in the access to data and inputs for current flow analysis methods; 3) to examine the scalability benefits of a cloud-based flow analysis tool.

CHAPTER 2: COUPLING SWAT AND MODFLOW MODELS FOR ENHANCED ASSESSMENT OF HYDROLOGIC PROCESSES AT THE WATERSHED SCALE

2.1: INTRODUCTION

In today's world of increased water demand in the face of population growth and climate change there are no simple solutions. For this reason many municipalities, water resource engineers, and federal analyses turn to modeling watersheds for a better understanding of the possible outcomes of their actions. These modeling efforts have typically focuses on either surface or subsurface water processes. The problem with this approach is that surface water models are typically unable to represent complex groundwater interaction and groundwater models lack the ability to model surface processes like plant growth. One popular surface water model, the Soil and Water Assessment Tool (SWAT) (Arnold et al, 1998) has repeatedly of performed poorly in heterogeneous groundwater driven systems (Peterson and Hamlett, 1998; Spruill et al., 2000; Chu and Shirmohammadi, 2004; Srivastava et al., 2006; Gassman et al., 2007). Similarly a popular groundwater model, modular groundwater flow (MODFLOW) (Harbaugh, 2005) is unable to simulate surface processes like overland flow, sheet erosion, channel erosion, plant growth, nutrient cycling, and agricultural management (fertilizer and pesticide application).

Due to the various benefits and drawbacks of watershed models there are a multitude available. A complication encountered when choosing a watershed scale model to use, is the multitude of choices. Many models are developed by research or government groups and are not typically used outside of those groups. A selection of some of the many available watershed scale models is graphed below in Figure 1 by how many peer-reviewed journal articles in a database refer to using the model. As illustrated below, the most popular models by far are the surface water model SWAT and the subsurface groundwater model MODFLOW, although TOPMODEL (Beven et al., 1995) is significantly more popular than some of the other models as well.

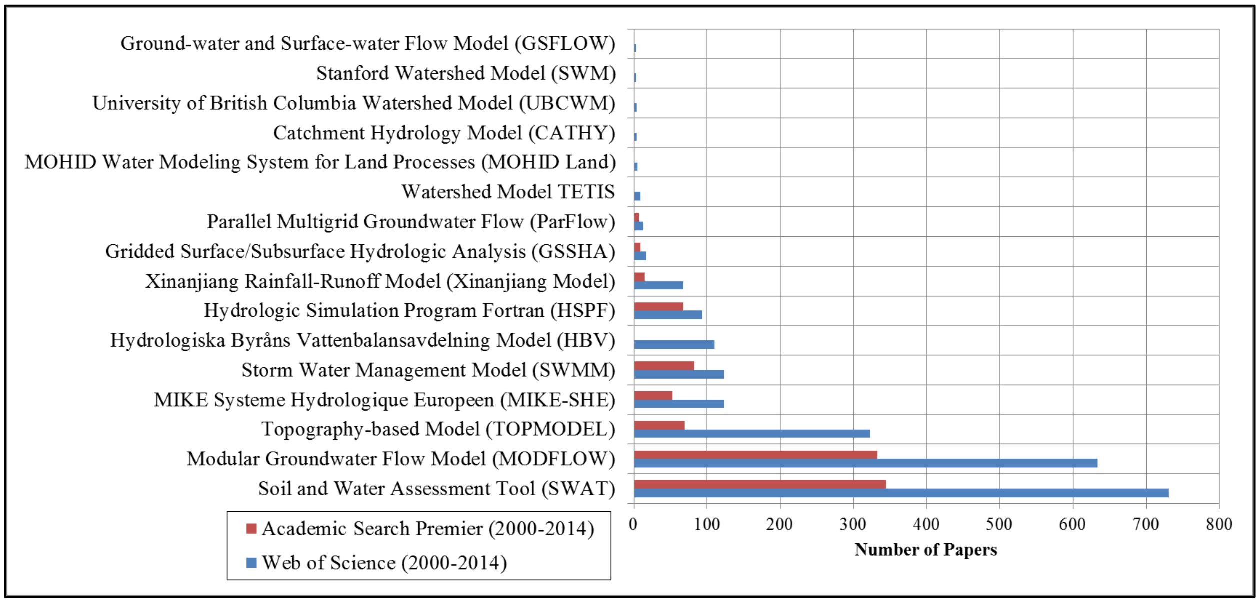


Figure 1: Watershed Models Listed by Number of Papers Available in Journal Paper Databases

SWAT has been a useful tool for assessing water resources, pollution problems, and assessing environmental conditions worldwide (Gassman et al., 2007). However, SWAT is a lumped watershed model with emphasis on plant and crop growth, nutrient cycling, and sediment yields from urban, natural, and agricultural areas. Using SWAT in a groundwater dominated system can sometimes fail to accurately represent the heterogeneous groundwater flow processes due to this lumped approach (Peterson and Hamlett, 1998; Spruill et al., 2000; Chu and Shirmohammadi, 2004; Srivastava et al., 2006; Gassman et al., 2007). For this reason there has been recent work to couple SWAT with the saturated finite-difference groundwater model MODFLOW. One of the earliest couplings was by adding a print statement to SWAT for parameters necessary for MODFLOW and then a read statement in MODFLOW to read them and use of a third software to convert the inputs for MODFLOW (Perkins and Sophocleous, 1999). A different combined use of SWAT and MODFLOW was used during a lake analysis by Menking (2003; 2004) in which a SWAT model's watershed outputs were used as inputs to the lake for a MODFLOW simulation. A more comprehensive simulation model was created by Galbiati et al. (2006) in which SWAT was coupled with MODFLOW for hydrologic simulation as well as MT3DMS for nutrient and chemical simulation. A later coupling of SWAT and MODFLOW via MODFLOW's river package (Harbaugh et al., 2010) was performed by Kim et al. (2008) and for a better representation of base-flow in a watershed in South Korea.

There are, however, numerous limitations to these existing couplings of SWAT and MODFLOW. The coupling of SWAT and MODFLOW by Perkins and Sophocleous (1999) relied on the use of a third software package to read and convert the SWAT outputs into MODFLOW inputs and vise-versa. The combination of SWAT and MODFLOW by Menking (2003; 2004) was not really a model coupling so much as using one model as an input preprocessor for the second model. The work by Galbiati that created a hydrology and nutrient/chemical coupling between SWAT, MODFLOW, and MT3DMS was only on a monthly basis thus restricting the interaction of SWAT and MODFLOW to only a monthly time step leaving sub-monthly groundwater interaction unrepresented. Kim et al. (2008) used a gridded preprocessing approach to force SWAT computational units (HRUs) to be the same size as the MODFLOW finite-difference grid cells, this reduces the computational efficiency of the lumped sub-basin approach that SWAT uses and requires the model extents and computational units to be the same size. This will typically require a user to create their own SWAT and MODFLOW project rather than using an available version of one model and having to only create the second.

The primary goal of this work is to enhance the capacity for modeling watershed processes by fully coupling surface and subsurface hydrologic processes at a daily time step. The specific objectives of this work are 1) to examine and create a general spatial linkage between SWAT and MODFLOW allowing the use of spatially-different existing models for coupling; 2) to examine existing practices and address current weaknesses for coupling of the SWAT and MODFLOW models to develop an integrated modeling system; 3) to demonstrate the capacity of the enhanced model compared to the original SWAT and MODFLOW models. The performance of the coupled SWAT-MODFLOW model will be tested in the North Fork of the Sprague River in the Upper Klamath Basin in Oregon due to the complex groundwater interactions, pre-existing spatially-dissimilar SWAT and MODFLOW models, and an interest the area's hydrology problems.

2.1.1: SWAT

The Soil and Water Assessment Tool (SWAT) was developed by the U.S. Department of Agriculture's Agricultural Research Service (USDA-ARS) (Arnold et al., 1998; Arnold and Fohrer, 2005; Neitsch et al., 2005; Gassman et al., 2007; Neitsch et al., 2011). It is a physically-based, basin-scale, pseudo-distributed, continuous-time watershed model emphasizing surface processes. SWAT operates by taking a single watershed, gauged or ungauged, and breaks it into multiple sub-basins which are then further broken into multiple unique combinations of land use, soil, and slope known as Hydrologic Response Units (HRUs). Calculations in SWAT are performed for each HRU and then scaled up to the sub-basin outlet by the percent area of the HRU within the sub-basin. This approach results in the HRUs lacking the spatial relations typically seen in a fully distributed model, but yields a computationally efficient calculation scheme allowing for watershed simulation over large periods of time.

Due to the long history of SWAT, almost 30 years, and the documentation of its subroutines, the application of SWAT models has grown worldwide in the past decade (Gassman et al., 2007). In the U.S., SWAT models are increasingly used to assist in Total Maximum Daily Load (TMDL) development (Borah et al., 2006) and to better understand the impacts of field management schemes for soil conservation and nutrient control. One meso-scale use of SWAT has been in the Hydrologic Unit Model of the U.S. (HUMUS) (Arnold et al., 1999b) in which all of the U.S. Geological Survey (USGS) 8-digit Hydrologic Cataloging Unit (HCU) watersheds (Seaber et al., 1987) in the continental U.S. were simulated. Use of the SWAT model however, has not been limited to the U.S., numerous projects in Europe have used SWAT to analyze and quantify the impacts of climate and management change on European watershed. One example being the European Commission's (EC) Climate Hydrochemistry and Economics of Surface-water Systems (CHESS) project (CHESS, 2001). The popularity and use of SWAT is apparent on its website database which contains over 1500 articles about SWAT used to examine watershed problems (https://www.card.iastate.edu/swat_articles). Additionally, to assist in the preparation of the various input files for the SWAT model, a free map-based ArcGIS interface call ArcSWAT was developed. ArcSWAT assists in delineating the watershed and processing raw data inputs like elevation maps and soil types into the required inputs files for the model.

The SWAT model begins with climate information daily precipitation, maximum and minimum temperature, solar radiation, relative humidity, wind speed. Then surface runoff is calculated by either the United States Department of Agriculture Natural Resources Conservation Service (USDA-NRCS) curve number method (USDA-NRCS, 2004) or the Green-Ampt method. Then a hydrologic balance, precipitation, interception, surface runoff, infiltration, evapotranspiration, lateral subsurface flow, and return flow from the shallow aquifer, per HRU is calculated and routed to the sub-basin's stream and then outlet. The land phase of the water balance calculated by SWAT per HRU is shown below in Figure 2 with units of mm of H_2O, where SW_t is the final soil water content, SW_0 is the initial soil water content, t is the time in days, R_{day} is the amount of precipitation on day i, Q_{surf} is the amount of surface runoff on day i, E_a is the amount of evapotranspiration on day i, w_{seep} is the amount of percolation and bypass flow exiting the soil profile bottom on day i, and Q_{gw} is the amount of return flow on day i.

$$\boxed{SW_t = SW_0 + \sum_{i=1}^{t}(R_{day} - Q_{surf} - E_a - w_{seep} - Q_{gw})}$$

Figure 2: SWAT Model Water Balance Equation

Evapotranspiration can be calculated using the Penman-Monteith (Monteith, 1965), Priestly-Taylor (Priestly and Taylor, 1972), or Hargreaves (Hargreaves et al., 1985) models (Gassman et al., 2007). SWAT also contains a number of subroutines to handle both forest and agricultural areas. Forest growth from seed to mature stand and crop planting, crop harvest, tillage, nutrient (fertilizer) application, and pesticide application can be simulated. The application amount and timing of fertilizers and pesticides can be customized using the many different management options allowed by SWAT. Sediment runoff from HRUs and channel erosion is also simulated in SWAT using the Modified Universal Soil Loss Equation (MUSLE) (Williams and Berndt, 1977). The nitrogen and phosphorous amounts in application and soil and water content can be simulated and tracked using multiple organic and inorganic nutrient pools and calculate a resulting nutrient loading to and concentration in the streams (Gassman et al., 2007). SWAT then lumps flow, sediment, nutrient, and pesticide and calculates loadings to the river with a lagged release, based on the time of concentration of the HRU. The scope of the processes covered by the SWAT subroutines is shown below in Figure 3; however the groundwater processes used by SWAT are a simplified lumped-parameter approach.

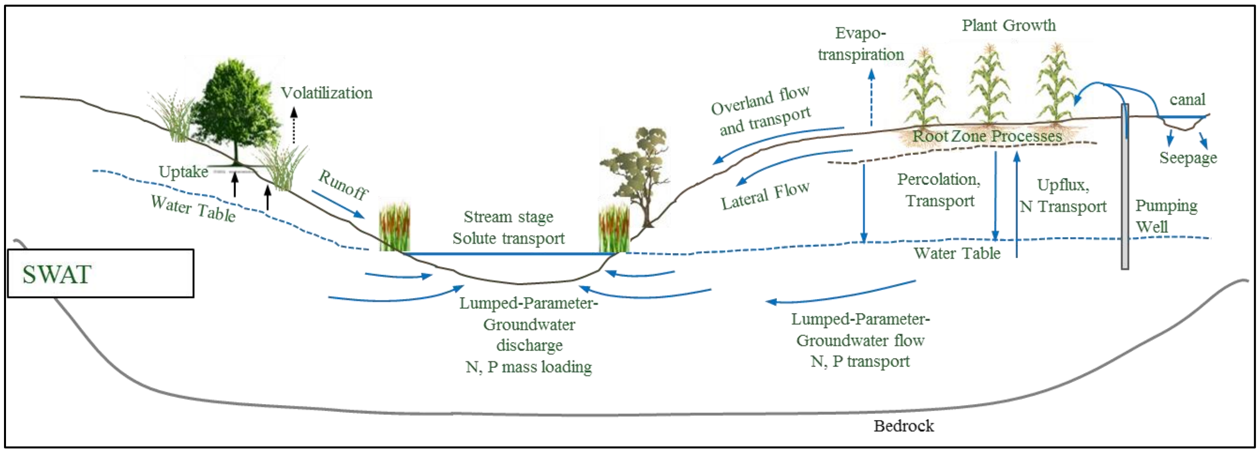


Figure 3: SWAT Model Watershed Processes

2.1.2: SWAT GROUNDWATER DISCHARGE TO RIVERS

As outlined by Gassman et al. (2007), there are numerous examples in which SWAT stream flow simulation has performed poorly for various groundwater-driven systems. SWAT contains groundwater subroutines but most of these problems stem from the lumped parameter approach used to handle what is actually a distributed groundwater flow processes. In northeast Pennsylvania, Peterson and Hamlett (1998) encountered problems applying SWAT to the Ariel Creek watershed for proper base-flow representation. They found that this complication was due to the presence of fragipan soils, soils containing a vertically impermeable layer causing more lateral flow. A further complication of not fully distributing the groundwater processes inside SWAT was discovered by Spruill et al. (2000) for a calibrated experimental watershed in which poor simulation of peak flows and recession rates were observed in combination with accurate monthly flows. A similar problem was found by Chu and Shirmohammadi (2004) with an unusually wet year in a 3.5 km^2 Maryland watershed. When the wet year was removed from the analysis period the model performed well; however, when included the model failed to properly estimate base-flow. Additionally, Srivastava (2006) found a poor representation of base-flow and other flow characterization by SWAT on the West Branch Brandywine Creek in Southwest Pennsylvania.

Some recent work on climate change impacts to wetland extent and water quality changes has focused on the Upper Klamath Basin in southwestern Oregon (Records, 2013). However, as illustrated by Gannett et al. (2010) the area of the Upper Klamath Basin is a heavily groundwater influenced system with an abundance of natural springs with complex interaction due to underlying volcanic strata. The sustained base-flow levels of the North Fork of the Sprague River have proven difficult to simulate using SWAT (Records, 2013) even after auto-calibration. Therefore, a manual calibration was then performed to fine tune the watershed parameters and yield a better match of base-flow on the North Fork of the Sprague River. The resulting simulation showed a more elevated base-flow closer to the real system but caused an annual trend in the base-flow level which was not observed in the watershed. For this reason it was determined that the lumped groundwater parameters of SWAT may be unable to handle complex groundwater processes like this without modifying the source code. A further discussion of SWAT’s inability to capture accurate groundwater processes in the Upper Klamath Basin is included in the Section 2.3: Results and Discussion.

2.1.3: MODFLOW

A similar watershed scale model to SWAT is the groundwater model MODFLOW developed by the United States Geological Survey (USGS) (McDonald and Harbaugh 1988; Hargaugh et al., 2000; and Harbaugh 2005). MODFLOW is a three-dimensional, saturated, physically-based, finite-difference groundwater model. This grid-based subsurface flow model combines a simple mass balance with Darcy’s law to simulate both steady and transient state groundwater conditions. A recent addition to the MODFLOW package is a Newtonian based solver algorithm which better satisfies the complex non-linear drying and re-wetting of grid cells in unconfined groundwater systems (Niswonger et al., 2011). Through the use of various packages in MODFLOW the surface and subsurface process of groundwater recharge (Harbaugh et al., 2000), vadose zone percolation (UZF1 package) (Niswonger et al., 2006), evapotranspiration (Harbaugh et al., 2000), and river-aquifer interactions (Harbaugh et al., 2000), and more can be simulated. Some of the various processes that can be modeled by MODFLOW are shown in Figure 4. An additional benefit to MODFLOW models is that numerous regional aquifers already have MODFLOW models built for them thanks to the work by the USGS (Christenson et al., 2011; Paschke 2011; Gannett et al., 2012; and Mashburn et al., 2013). However, MODFLOW is unable to simulate overland flow, sheet erosion, channel erosion, plant and crop growth, nutrient cycling, or agricultural management (pesticides and fertilizers).

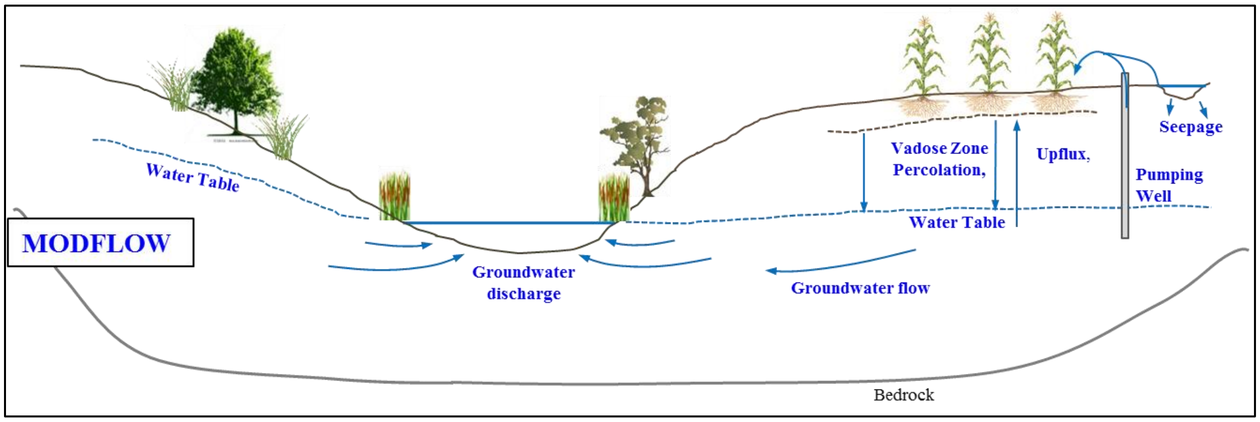


Figure 4: MODFLOW Model Watershed Processes

Like most groundwater models, MODFLOW lacks the ability to simulate pollutant transport because it focuses only on groundwater hydrology. For this reason there have been a number of efforts to model pollutant advection, dispersion, and reaction based on the outputs of common groundwater models like MODFLOW. One such modular three-dimensional transport model (MT3D) was developed by Zheng (1990). MT3D was built to use MODFLOW's groundwater hydrology results and determine a pollutant reactive transport solution; later MT3D was modified into MT3DMS to handle multi-species transport and interaction (Zheng and Wang, 1999; Prommer et al., 2003). Another similar model for using MODFLOW results to model pollutant transport is the Reactive Transport 3D model (RT3D) developed by Clement et al. (1998). RT3D was later modified to handle 1-dimensional unsaturated groundwater pollutant transport using the results of the MODFLOW-UZF1 package (UZF-RT3D) (Bailey et al., 2013b). However, a continued issue with this methodology is that these packages are not built into MODFLOW and only simulate subsurface chemical species transport. Although this is a step in the direction towards a complete watershed model, even with the use of RT3D or MT3DMS, MODFLOW is still unable to handle surface processes, channel erosion, and plant growth/cycling due to its limitation to subsurface processes.

2.1.4: SURFACE-SUBSURFACE WATERSHED MODELING

The idea of enhancing surface water models to more accurately reflect groundwater or groundwater models to more accurately reflect surface water processes is nothing new, but there are numerous ways to approach it. One combination of SWAT with a distributed groundwater model was by Perkins and Sophocleous (1999; Sophocleous and Perkins 2000). In this case, SWAT was combined with MODFLOW in order to pass in the MODFLOW inputs of tributary flow, recharge, and evapotranspiration (ET). This model was built, calibrated and tested in a set of river basins in Kansas to facilitate a better understanding and analysis of the groundwater pumping in the area. The coupling between these models was accomplished with a read/write subroutine added to both SWAT and MODFLOW for input files while an interceptor program was written to handle the conversion of the different spatial and temporal scales between models.

MODFLOW was modified to include an additional subroutine which took SWAT tributary flow results and initialized and mapped stream outputs to grid cells and ET, recharge and pumping/irrigation demands. SWAT was similarly modified to average land use and HRU differences per sub-basin to pass into the MODFLOW model. SWAT was further modified to lump and output simulation results per day, month, or year for use with MODFLOW's groundwater modeling time step.

A continuation and addition to this model was performed by Conan et al. (2003) with the addition of a groundwater nutrient transport model to the coupling. The watershed of interest this time was the Coet-Dan watershed in Brittany, France where a nitrate problem has developed due to historical agriculture development. A preliminary investigation of the pollutant led to the understanding that the nitrate surface pollutant leached to the groundwater and was then transported to the streams by groundwater base-flow. This combined surface water/groundwater interaction lent itself to being modeled by the combined surface water/groundwater model of Sophocleous's SWAT and MODFLOW model. The only complication was the lack of a groundwater nutrient model. The groundwater pollutant advection, dispersion, reaction model, MT3D (Zheng, 1990) was therefore incorporated into the combined SWAT and MODFLOW model to handle unsaturated zone flow nitrate leaching and complete the watershed model in both water quantity and quality. Again, this coupling of SWAT and MODFLOW was accomplished through the use of a third input conversion model outside of the SWAT and MODFLOW codes.

A combined use, not combined model, of SWAT and MODFLOW was performed on the Estancia basin in New Mexico (Menking et al., 2003) and a look into the same basin during the last glacial period was also examined (Menking et al., 2004). In these studies a SWAT model was used to determine overland hydrology inputs to a large lake and the lake level was then modeled using MODFLOW with the lake (LAK2) package (Council, 1999). The SWAT and MODFLOW models were not coupled so much as compared and various inputs/outputs were post processed and used in the other model.

Another surface water-groundwater model combination, ParFlow, by Kollet and Maxwell (2006) incorporates a two dimensional surface model with a three dimensional variably saturated groundwater flow model. The surface model simulates land surface, vegetation, and overland flow and is used as a boundary condition to the groundwater model allowing a simultaneous solution of the two systems. The resulting surface/subsurface model compared well to the existing models of Hec-1, MODHMS, and Hydrologic Simulation Program-Fortran (HSPF). While the surface model in ParFlow allows for calculation of evaporation, transpiration, freeze-thaw processes, and heat fluxes (Ferguson and Maxwell, 2010) it does not simulate nutrient cycling and erosion yields.

A different combined surface/subsurface watershed model was created by Galbiati et al. (2006) where they combined SWAT, MODFLOW, and MT3DMS into the Integrated Surface-Subsurface Model (ISSM). In this combination SWAT was used to simulate the surface water dynamics as well as unsaturated zone and plant interactions while MODFLOW was used to simulate the saturated zone hydrology. Then MT3DMS was used to determine pollutant advection, dispersion, and reaction in the subsurface and pass the results back to SWAT. The coupling of SWAT and MODFLOW in ISSM was accomplished through the stream routing (STR1) package of MODFLOW (Prudic et al., 2004). SWAT and MODFLOW were run on a monthly time step with stream seepage and base-flow contribution from MODFLOW added to the results of SWAT’s own stream routing functions. Combining the model in this fashion and calibrating the resulting model for the Bonello watershed in Italy yielded reasonable results on a heavily agricultural and costal watershed. This monthly coupling however does not lend itself to simulation of a complex sub-monthly groundwater-surface water connection because the models are only run on a monthly time step.

An early attempt to augment the algorithms in MODFLOW to better represent surface processes resulted in the creation GSFLOW (Markstrom et al., 2008). GSFLOW combines the groundwater model of MODFLOW-2005 with the Precipitation Runoff Model System (PRMS). GSFLOW couples the models to simulate the surface domain and governing equations independently of the subsurface domain. This coupling is different than the approach taken by ParFlow (Kollet and Maxwell, 2006) which simultaneously solves the surface and subsurface governing equations. GSFLOW can provide an accurate simulation of the hydrologic response in both the surface and subsurface systems but still lacks the ability to simulate nutrient cycling, pesticide transport, crop-growth and die-off, or sediment yields.

Work by Kim et al. (2008) took a gridded approach to SWAT and chose to combine it with the finite-grid groundwater model MODFLOW. Kim et al. preprocessed the SWAT input information into the same grid sizes used by the MODFLOW model so that the passing of information from grid-based HRUs to grid-based MODFLOW could be accomplished without multiple HRUs contributing to a grid or vise-versa. The aquifer-river interactions in this particular model were handled by MODFLOW using the river (RIVR) package (Harbaugh et al., 2010). The resulting model was applied to the Musimcheon Basin in Korea resulting in a better representation of stream base-flows and groundwater pumping effects than the original SWAT method. A continuation of the work by Kim et al. (2008) takes a special look at groundwater recharge rates in the Mihocheon watershed in South Korea (Chung et al., 2010) and found them much better represented in the combined SWAT-MODFLOW model than by the original SWAT alone. This coupling was more closely coupled than the work by Sophocleous and Perkins (1999) but still fails to retain the full computational benefits of the pseudo-distributed original SWAT model and uses extra computer resources to read and write the input files to pass between the two models.

Another combined surface-subsurface hydrology model, Catchment Hydrology (CATHY) uses a path-based overland flow combined with a coarser gridded subsurface model (Camporese et al., 2010). CATHY uses threshold based boundary conditions to convert potential water fluxes in the system into actual fluxes passed between the surface and subsurface modules within it. CATHY includes routines for hill slope runoff, channel flow, lake areas, and subsurface interactions. However, CATHY is not capable of simulating plant growth or nutrient transport/cycling because it is only a hydrology model.

A further continuation of the work by Sophocleous and Perkins (1999) is a special modification of their combined SWAT-MODFLOW model (SWATMOD99) to better handle unsaturated and saturated zone processes in an arid environment (Luo and Sophocleous, 2011). They added special subroutine process to handle the conversion and pass back of groundwater depth, percolation, and evapotranspiration. After modification this model was applied to the Hetao Irrigation District, Inner Mongolia, China and illuminated high conveyance losses in the irrigation canals of the area. The issue still with this model is the loose coupling of the systems and as well as the outdated SWAT and MODFLOW software versions. To partially address this, Luo and Sophocleous (2011) updated this coupling from SWAT 99.2 to SWAT 2000 but the MODFLOW model remained the 1996 version. It would be more beneficial for this model to be combined with the newest available version of SWAT, SWAT 2012 (Neitsch et al., 2011) and the most recent MODFLOW, MODFLOW-NWT (Niswonger et al., 2011).

Because SWAT is a pseudo-distributed model a logical step to better represent the heterogeneous distributed that actually exists would be to disaggregate SWAT's HRUs in a preprocessing to support a more gridded approach. This in turn can better represent groundwater properties and processes in the same model by reflecting the spatial heterogeneity of the system in the inputs. One example of this is the work by Rathjens and Oppelt (2012), in order to retain the high resolution spatial inputs they built a gridded SWAT setup. A gridded approach is vastly different than the sub-basin/HRU setup currently in SWAT. This is primarily evident in that the gridded runoff needs to be routed from one grid to the next and eventually to the river as opposed to the current SWAT infrastructure which routes each HRU independently to the sub-basin's river then to the next sub-basin. The end result of the modifications was titled SWATgrid and successfully tested on the Lake Fork Texas watershed in Texas. This is an attempt to make the existing subroutines of SWAT handle the distributed processes of groundwater flow and transport perform better. However the existing SWAT routines are unable to handle complex groundwater processes and groundwater pollutant transport problems like the reactive transport of selenium and nitrate in an irrigated agricultural area (Bailey et al., 2013a; Bailey et al., 2014).

2.2: METHODS AND MODELS

2.2.1 OVERVIEW

The Soil and Water Assessment Tool (SWAT) (Arnold et al., 1995b; Arnold et al., 1998), and the U.S. Geological Survey's Modular Ground-Water Flow Model (MODFLOW) (Harbaugh, 2005) were chosen for coupling. SWAT has been a useful tool for assessing water resources, pollution problems, and assessing environmental conditions worldwide (Gassman et al., 2007). In addition to MODFLOW's popularity, there are many regional-scale models available in the U.S. (Rumman and Payne 2003; Christenson et al. 2011; Paschke, 2011; Gannett et al., 2012; Mashburn et al., 2013). Summarized herein are the use of the Upper Klamath Basin MODFLOW model (Gannett et al., 2012) and a SWAT model for the Sprague River (Records, 2013). The SWAT and MODFLOW models were coupled on a daily time step to allow for greater model feedback. Both the entire Sprague River and the groundwater influenced North Fork were chosen as a comparison to the original SWAT model and MODFLOW model. The SWAT model was calibrated by Records (2013) for the period from 2001 to 2006 while the MODFLOW model was calibrated by Gannett et al. (2012) for the period from 1970 to 2004. The period of analysis for the combined SWAT-MODFLOW model is from 1995 to 2004, which is the entire period of overlap between the SWAT, MODFLOW, and observed data. USGS Station 11495800 was used for comparison of the North Fork of the Sprague River and the USGS Station 11501000, prior to the Sprague River's confluence with the Williamson River, was used for comparison of the entire Sprague River watershed.

2.2.2 MODEL COUPLING

In order to create a comprehensive surface/subsurface watershed model the surface model SWAT 2012 Revision 591 (Arnold et al., 1998) was coupled with the latest version of the groundwater model MODFLOW-NWT (Niswonger et al., 2011), which contains a Newtonian based solution method to analyze the non-linear drying/rewetting of grid cells for the original MODFLOW-2005 (Harbaugh, 2005). These models were coupled in such a fashion as to retain their respective strengths; SWAT was allowed to handle land surface processes, in-stream processes, and SWAT's "soil zone" processes; MODFLOW-NWT handles inputs to the aquifer with either 1-dimensional unsaturated subsurface recharge, using the Unsaturated Zone Flow (UZF1) package (Niswonger et al., 2006) or aquifer recharge with MODFLOW’s Recharge (RCH) package (Harbaugh et al., 2000), interaction with the stream network, and groundwater pumping. Figure 5 and Table 1 outline each watershed process or concept as handled by either SWAT or MODFLOW in the coupled SWAT-MODFLOW (SM) model.

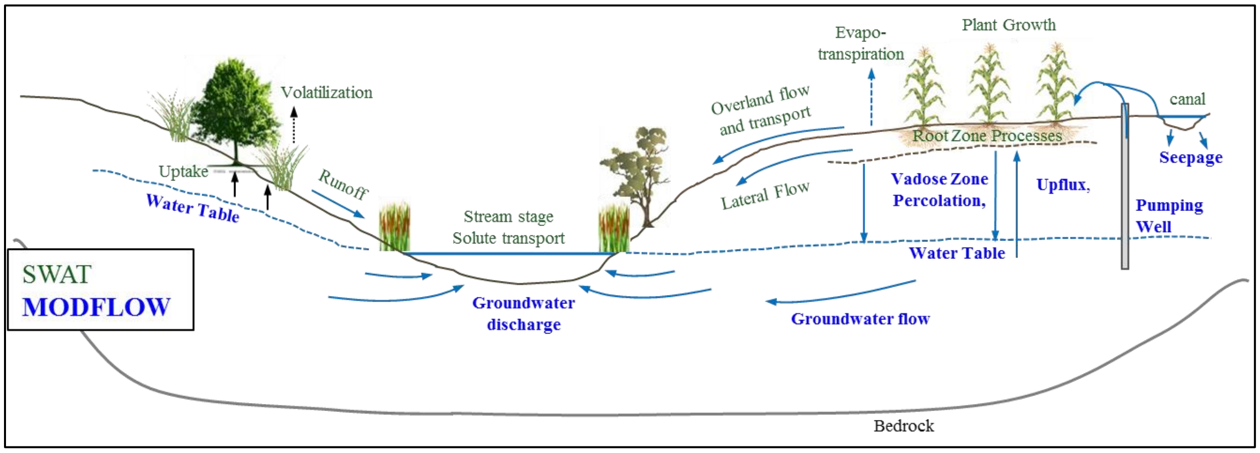


Figure 5: SWAT-MODFLOW Coupled Model Watershed Processes

Table 1: SWAT-MODFLOW Coupled Model Watershed Processes

SWAT Simulation	MODFLOW Simulation	
Infiltration	Aquifer Recharge via Either:	The RCH package (Harbaugh et al., 2000)
Evapotranspiration		Vadose zone percolation below the soil profile via the UZF1 package (Niswonger et al., 2006)
Plant Growth and Root Zone	Water Table Elevation	
Overland Flow and Transport	Saturated Groundwater Flow	
Lateral Subsurface flow in SWAT’s “Soil Zone”	Groundwater pumping, via the WEL package (Harbaugh et al., 2000)	
Stream Flow and Transport	Groundwater discharge to streams (base-flow), stream seepage to groundwater, via the RIVR package (Harbaugh et al., 2000).	

To facilitate a more general coupling between SWAT's spatially discontinuous HRU variables and MODFLOW's spatially continuous grid-based variables, a series of linking functions to convert the variables were developed. This information was preprocessed using the geospatial software (ArcGIS Desktop: Release 10, Environmental Systems Research Institute Redlands, CA). Full step by step creation of the necessary inputs for this coupled SWAT-MODFLOW model is documented in Appendix I. As a short explanation, the HRUs of SWAT were spatially disaggregated and then intersected with the MODFLOW grid. The intersection provides information on what percentage of an HRU contributes to which grid and vise-versa. Using these percent areas as weights for averaging, the various linking variables are mapped from SWAT HRUs to MODFLOW grid cells and back, illustrated in Figure 6. This general contributing area approach allows for SWAT HRUs larger in size than MODFLOW grid cells or grid cells larger than HRUs. Additionally, all of the subroutines modified were checked to allow a MODFLOW model with greater in spatial coverage than the SWAT model or vise-versa to be combined in the same fashion.

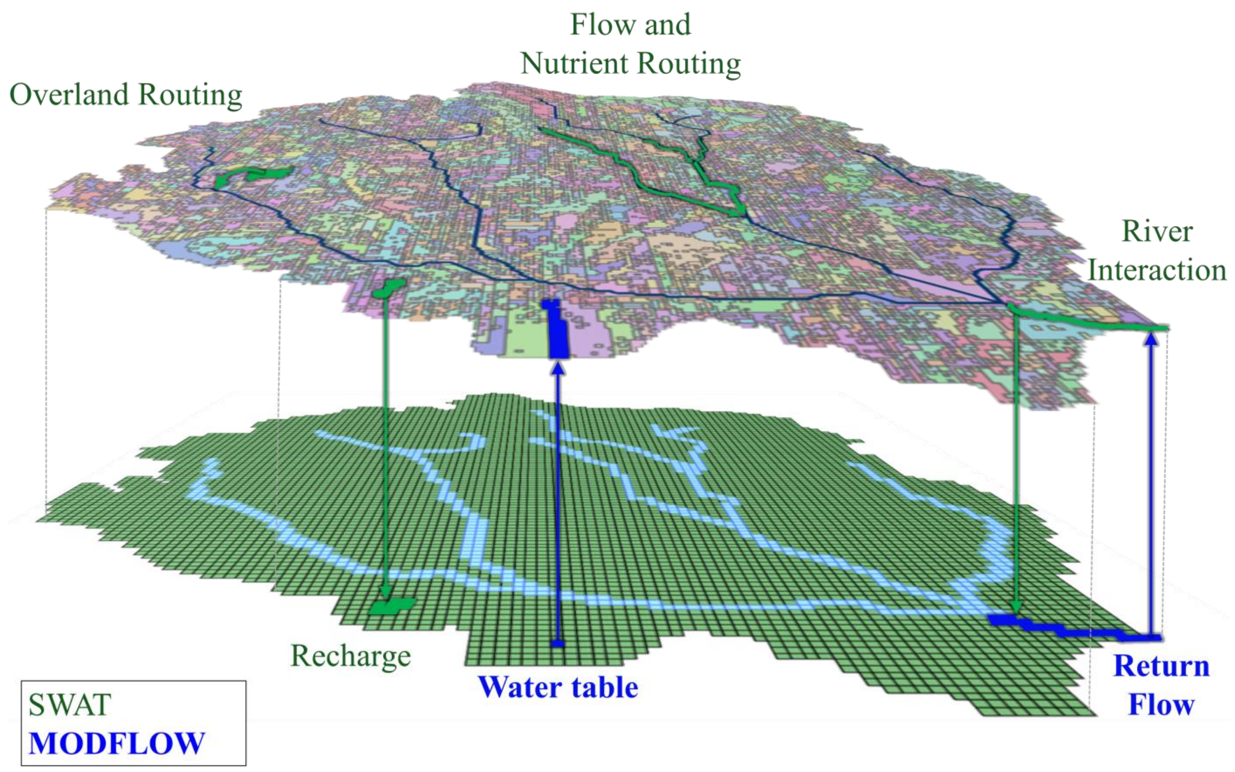
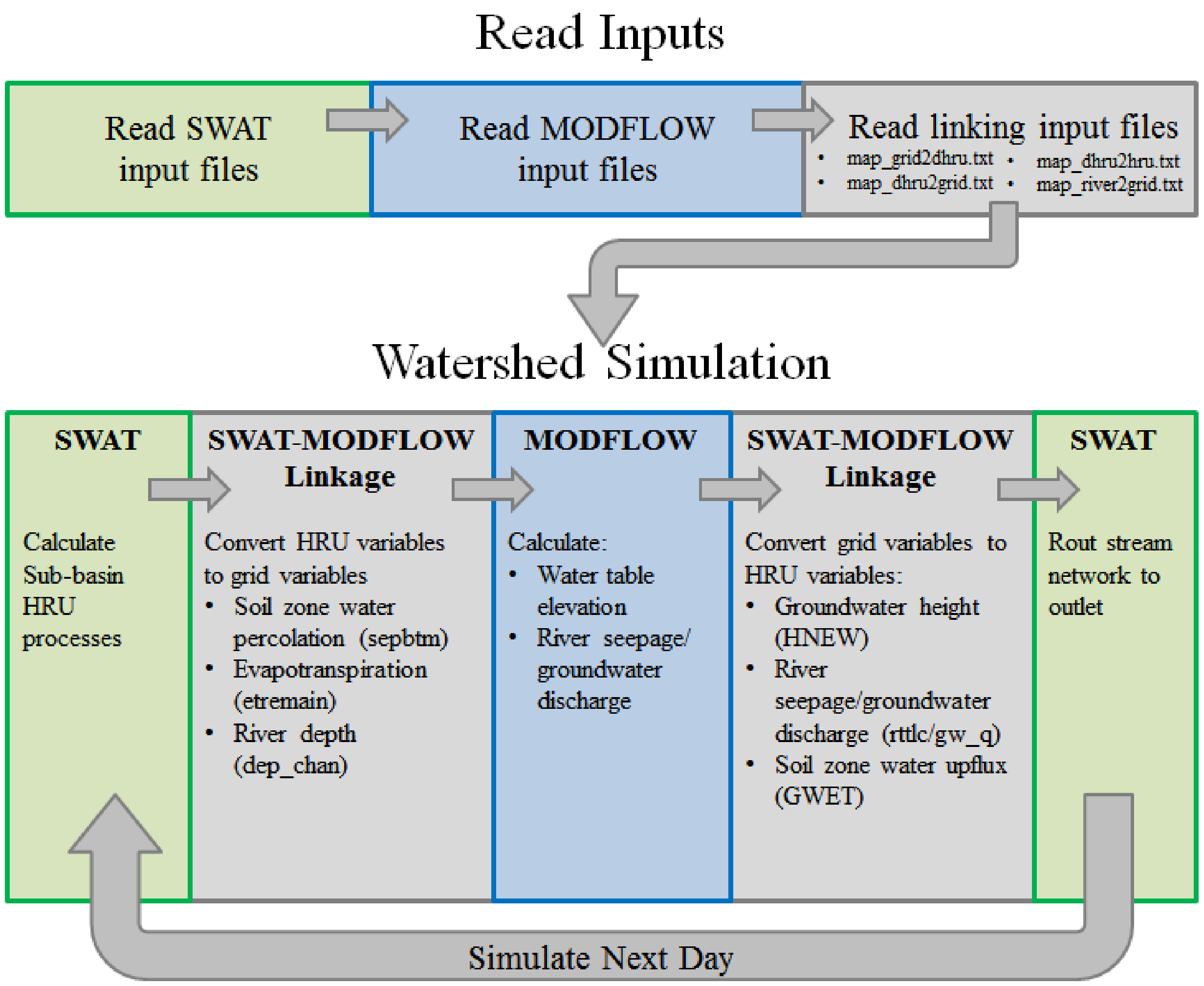


Figure 6: SWAT-MOFLOW Model Spatial Interaction; SWAT (green text), MODFLOW (bold blue text)

In order to process the information required for these conversions, output from the GIS operations, a series of preprocessing scripts were written in Java, a copy of this code is available in Appendix II. The resultant output files from the Java pre-processing summarize which HRUs contribute what percent are of each grid, and vise-versa. To facilitate the conversion from HRUs to grids, there are 4 primary input files. The first, map_dhru2hru.txt, summarizes which spatially-disaggregated HRUs (DHRUs) contribute to each of the original SWAT HRUs; this step allows groundwater processes to remain fully distributed in MODFLOW. The second linking file, map_dhru2grid.txt, summarizes which DHRUs contribute to the MODFLOW grid cells. Map_grid2dhru.txt similarly summarizes which MODFLOW grid cells contribute to each DHRU for conversion of MODFLOW information back to SWAT. The final linking file, map_river2grid.txt, provides the necessary information to convert SWAT river information into MODFLOW river grid cell information. An example of the template and explanation of each of these linking files is available in Appendix III. Similarly, each of the FORTRAN conversion subroutines used to couple these models are available in Appendix IV. To assist with a general linkage of spatially-different SWAT and MODFLOW models, these functions were written to change information only for HRUs that intersect grids and vise-versa. To increase model coupling compared to previous SWAT-MODFLOW integrations in an attempt to capture sub-monthly groundwater-surface water interaction, it was decided to execute MODFLOW on a daily time step. The general flow of information on a daily basis is outline in Figure 7.



Read Inputs
Read SWAT input files
Read MODFLOW input files
Read linking input files
map_grid2dhru.txt
map_dhru2grid.txt
map_dhru2hru.txt
map_river2grid.txt
Watershed Simulation
SWAT
Calculate Sub-basin HRU processes
SWAT-MODFLOW Linkage
Convert HRU variables to grid variables
Soil zone water percolation (sepbtm)
Evapotranspiration (etremain)
River depth (dep_chan)
MODFLOW
Calculate:
Water table elevation
River seepage/ groundwater discharge
SWAT-MODFLOW Linkage
Convert grid variables to HRU variables:
Groundwater height (HNEW)
River seepage/groundwater discharge (rttlc/gw_q)
Soil zone water upflux (GWET)
SWAT
Rout stream network to outlet
Simulate Next Day

Figure 7: SWAT-MODFLOW Code Process Diagram

When the coupled model runs, it begins by reading in all of the SWAT input files as normal. Once the SWAT input files are read in, the MODFLOW input files and linking files are also read in, Appendix IV: sm_read_dhru2grid, sm_read_dhru2hru, sm_read_grid2dhru, sm_read_river2grid. Then SWAT executes normally through all of its sub-basin and HRU calculations. Once SWAT finished calculating its HRU processes, a subroutine to prepare input variables from SWAT to MODFLOW is called Appendix IV: sm_conversion2mf. This subroutine was written to convert the SWAT HRU variables and units into spatially located DHRU linkage variables ready to be converted inside of MODFLOW. Specifically, this step converts SWAT HRU-based variables of percolation from the bottom of the soil profile (sepbtm) and remaining evapotranspiration (etremain) into DHRUs. The array variable etremain was added to SWAT to track the remaining unsatisfied evapotranspiration (potential ET minus actual ET in SWAT's subbasin subroutine) which is needed as an input for MODFLOW. It is also in this step that SWAT units are converted into the units of the current MODFLOW model. This unit conversion step is required because in general SWAT runs on SI units whereas MODFLOW can use a variety of units as specified by the current model inputs; a MODFLOW model can run with units of feet and days, meters and months, centimeters and seconds, or others. Again, this general coupling is to support the use of existing dis-similar spatial scales and units of SWAT and MODFLOW models.

In the next step of the code MODFLOW is called. As MODFLOW executes it checks which of its various packages are active and if a new stress period has been reached, it reads in the required information for each active package. If the river (RIVR), recharge (RECH), or unsaturated zone flow (UZF1) packages are active, it then a linkage subroutine is called to convert the SWAT variables into the MODFLOW variables, Appendix IV: sm_mfRiver, sm_recharge, and sm_uzf. In this coupling, the primary linkage between the rivers of SWAT and MODFLOW is handled using MODFLOW's RIVR package (Harbaugh et al., 2000). Furthermore, the original MODFLOW-NWT model was modified to include a day counter, forcing MODFLOW to execute on a daily time step. This modification includes a reader-reset that allows MODFLOW to read in new stress period information, which may or may not be on a daily time step, as it normally does when the SWAT-MODFLOW simulation reaches the next of MODFLOW stress period. Once MODFLOW is done executing for the day, another conversion subroutine is called to pass information back to SWAT, Appendix IV: sm_conversion2swat. This subroutine finds and converts the water table elevation from MODFLOW grids and units to SWAT HRUs and units. It also passes back the MODFLOW determined stream loss/gain per grid cell to SWAT river variables. Stream gain, or groundwater discharge (SWAT's gw_q variable), is passed back based on relative area of each HRU contributing to a given river segment. Stream loss, or seepage, is passed back to the sub-basin as a whole to SWAT's reach loss variable, rttlc, which was changed to a global array instead of a global value to allow tracking of this information. As a final step, Appendix IV: sm_upflux_to_soil, in locations where the water table has reached the SWAT's soil zone, upflux water is passed back water into the soil zone based on soil wilting point and field capacity. A summary of these daily interactions is shown in Figure 8.

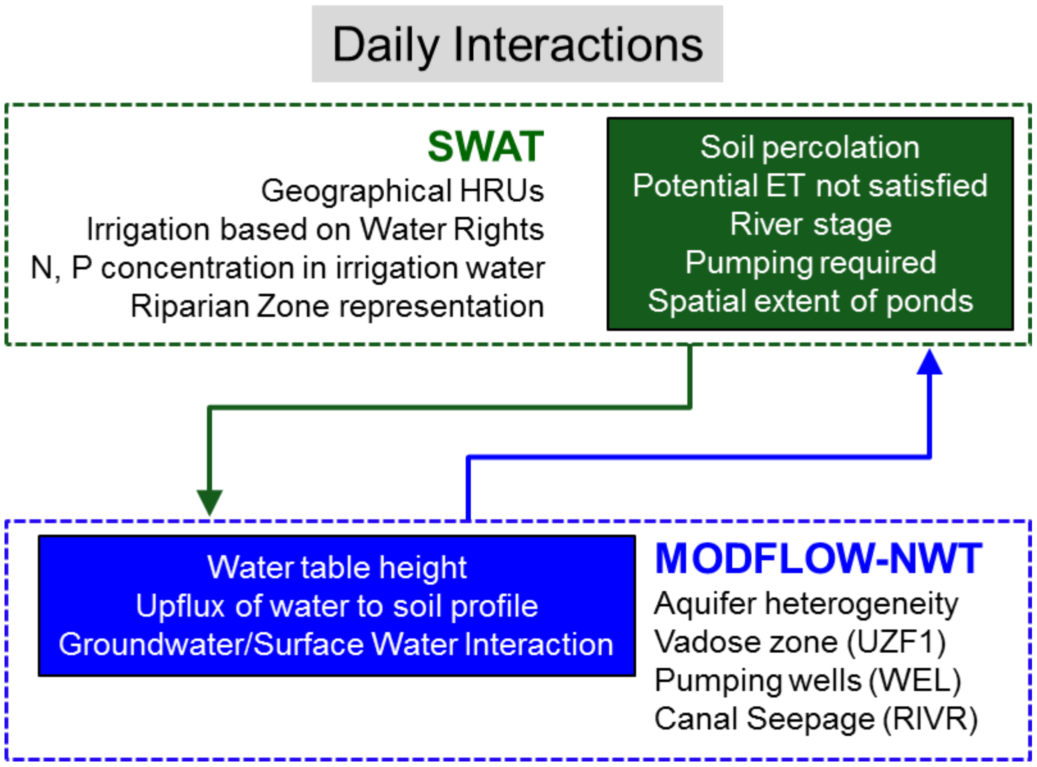


Figure 8: SWAT-MOFLOW Coupled Model Information Flow Diagram

Once the information from MODFLOW is passed back into SWAT, the normal SWAT river routing subroutines are called to transport water, sediment, and nutrients to the basin outlet and the model progresses to the next day of simulation. All of these steps take place in memory within the FORTRAN code resulting in a single program rather than a SWAT model, a MODFLOW model, and a coupling model as three programs.

2.2.3: APPLICATION

The application of this new style of SWAT-MODFLOW coupling was applied to the Upper Klamath Basin in Oregon, Figure 9. An analysis of both the North Fork of the Sprague River and the entire Sprague River watershed was performed.

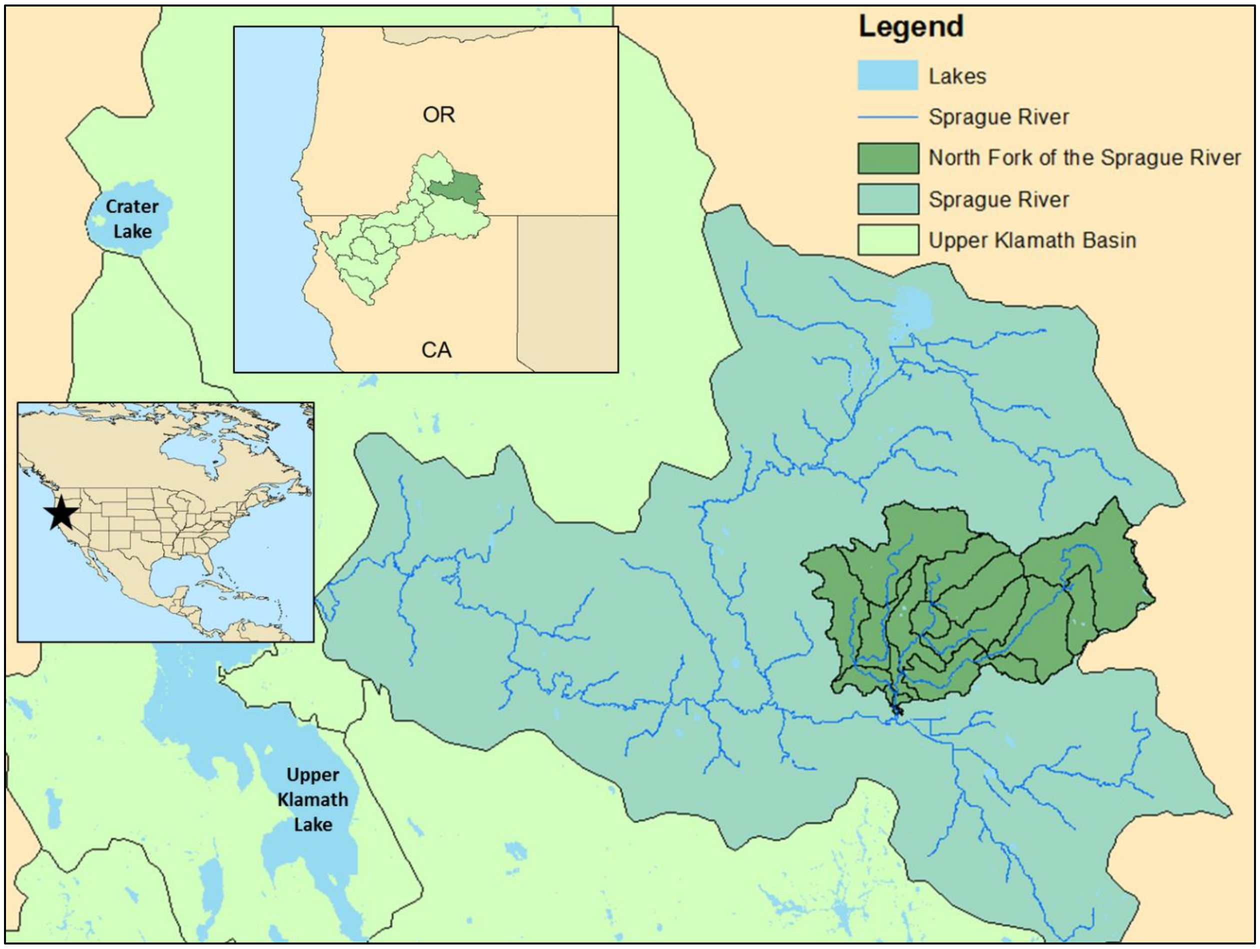


Figure 9: Sprague River Watershed Location, Upper Klamath River Basin, OR

The Upper Klamath Basin has been highlighted for its key groundwater influence on surface water hydrology reflecting a complex groundwater-driven watershed (Gannett et al., 2010). It has also been the focus of historic and recent intensive agricultural projects, specifically recently the U.S. Bureau of Reclamation's Klamath Irrigation Project. Due to regional interest, a MODFLOW groundwater model was developed by the USGS for the entire Upper Klamath Basin (Gannett et al., 2012) to assess the potential impacts of increased groundwater development with in the basin, primarily for agricultural use. An overview of the MODFLOW model combined with the SWAT models, the North Fork of the Sprague River and the entire Sprague River, is shown below in Figure 10.

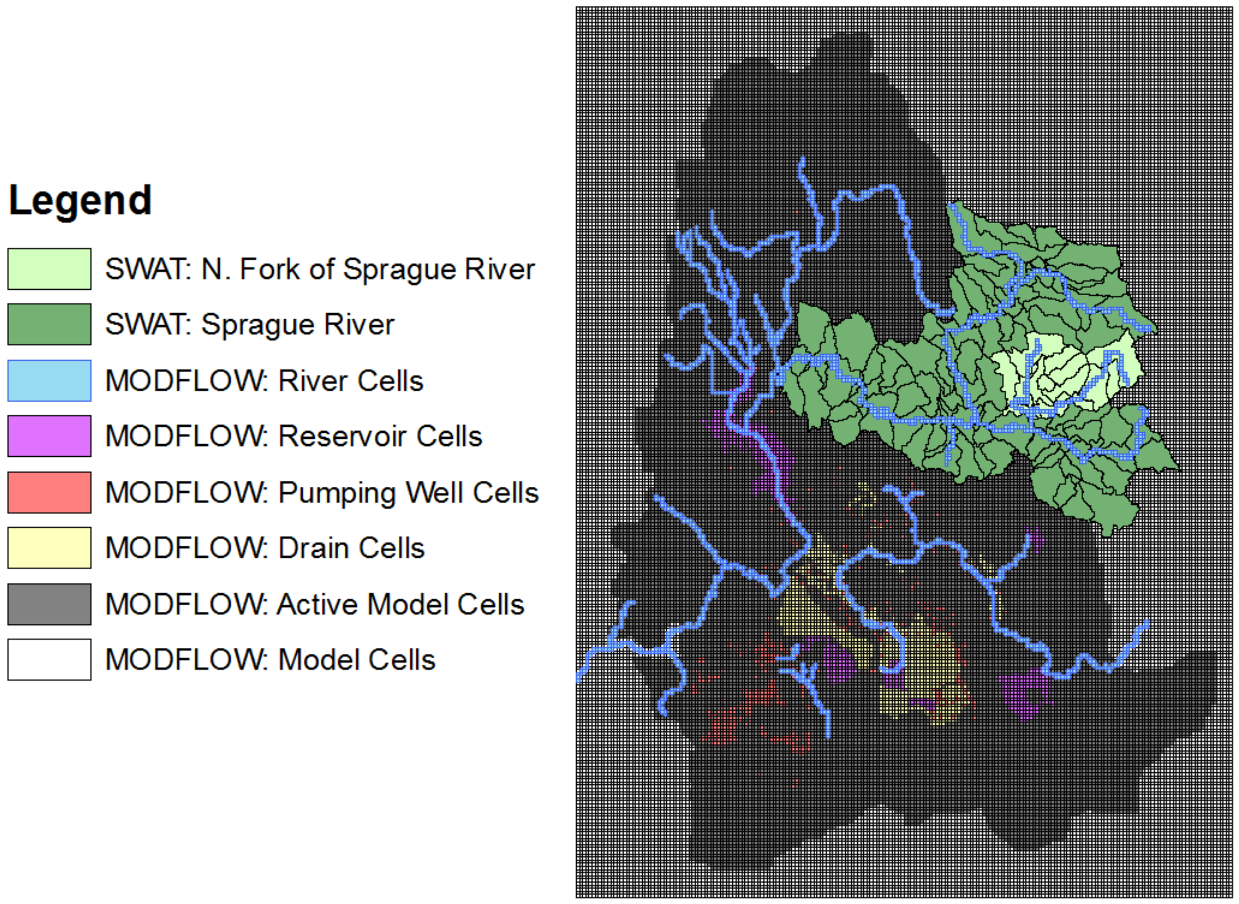


Figure 10: SWAT Model's Location Within USGS's Upper Klamath Basin MODFLOW Model

Furthermore, Records (2013) studied the impacts to the extent and function of wetlands under various climate predictions. As a result of this work a SWAT model was developed for the Sprague River basin. The SWAT model performed well for surface process dominated tributaries like the Sycan River, Figure 11, but performed poorly for groundwater driven tributaries like the North Fork of the Sprague River, Figure 12, even after auto calibration and manual calibration. For these reasons, the new style of coupling of SWAT and MODFLOW was chosen to be demonstrated on the North Fork of the Sprague River with an overlap of the SWAT model developed by Records (2013) and the MODFLOW model developed by Gannett et al. (2012), an overview of these SWAT and MODFLOW model extents and locations are shown in Figure 10.

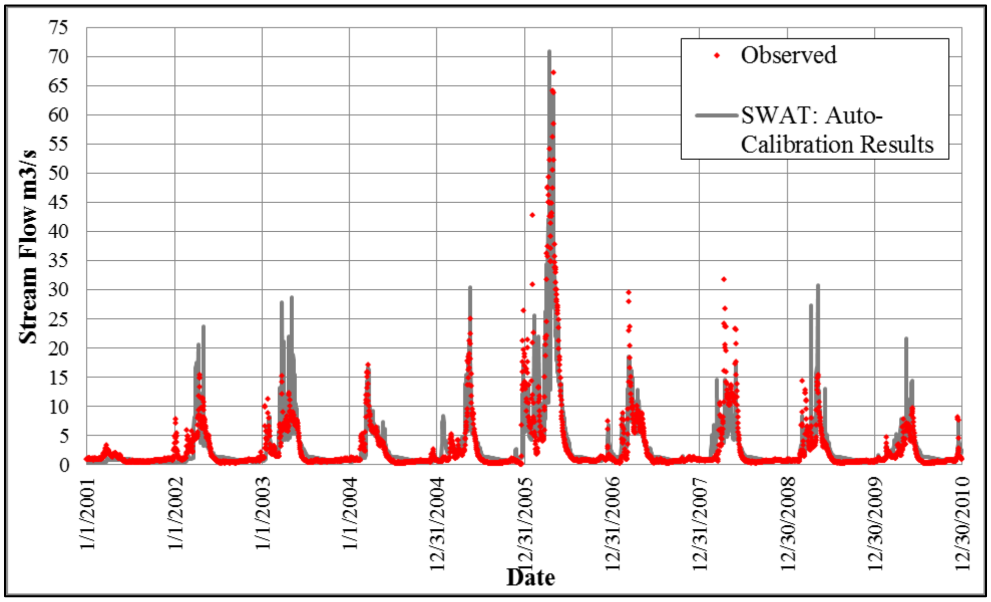


Figure 11: SWAT Auto-Calibration Results for the Sycan River, OR

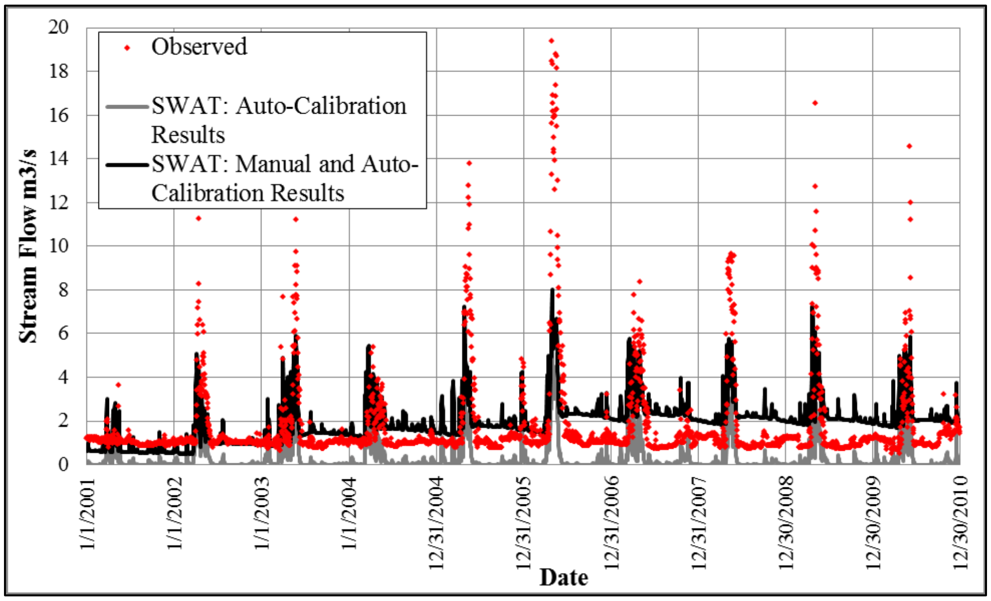


Figure 12: SWAT Combined Auto-Calibration and Manual Calibration Results for the North Fork of the Sprague River, OR

2.3: RESULTS AND DISCUSSION

An examination of the results of the SWAT-MODFLOW model coupling is contained herein for both the groundwater influenced North Fork of the Sprague River and the entire Sprague River.

2.3.1: NORTH FORK OF THE SPRAGUE RIVER

2.3.1.1: STREAM FLOW RESULTS

An initial examination of the results of the combined SWAT-MODFLOW model for the North Fork of the Sprague River reveals a lower base-flow than observed but still a relatively good fit to observed data (shown on a monthly time-step in Figure 13). A 1-to-1 comparison of SWAT and SWAT-MODFLOW versus observed daily data is shown in Figure 14 and a similar 1-to-1 comparison for monthly data is shown in Figure 15. Statistics summaries for the fit of the models to daily and monthly observed data are presented in Table 2 and Table 3, respectively. Both the daily and monthly comparisons show a slight improvement in coefficient of determination (R^2) and poorer, but still acceptable, Nash-Sutcliffe (NS) coefficients of performance, between the original SWAT model (daily R^2 = 0.35, NS = 0.23; monthly R^2 = 0.60, NS = 0.57) and the SWAT-MODFLOW model (daily R^2 = 0.34, NS = 0.14; monthly R^2 = 0.66, NS = 0.38). The original Sprague River and North Fork of the Sprague River SWAT models were calibrated on a monthly basis while the original Upper Klamath MODFLOW model was calibrated using quarterly stress periods, which explains the improvement between daily and monthly statistics. The larger time-step of the original calibration is the likely source of the poorer daily performance statistics for the coupled SWAT-MODFLOW model. Additional calibration using a sub-monthly time-step may improve the simulation results.

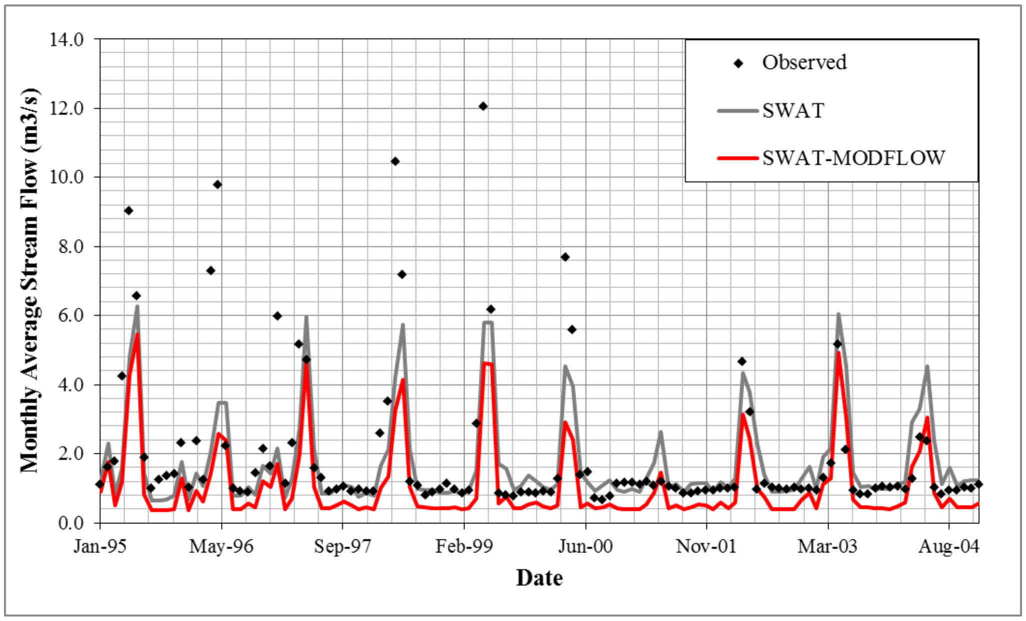


Figure 13: SWAT-MODFLOW Monthly Results, No Additional Calibration for the North Fork of the Sprague River OR

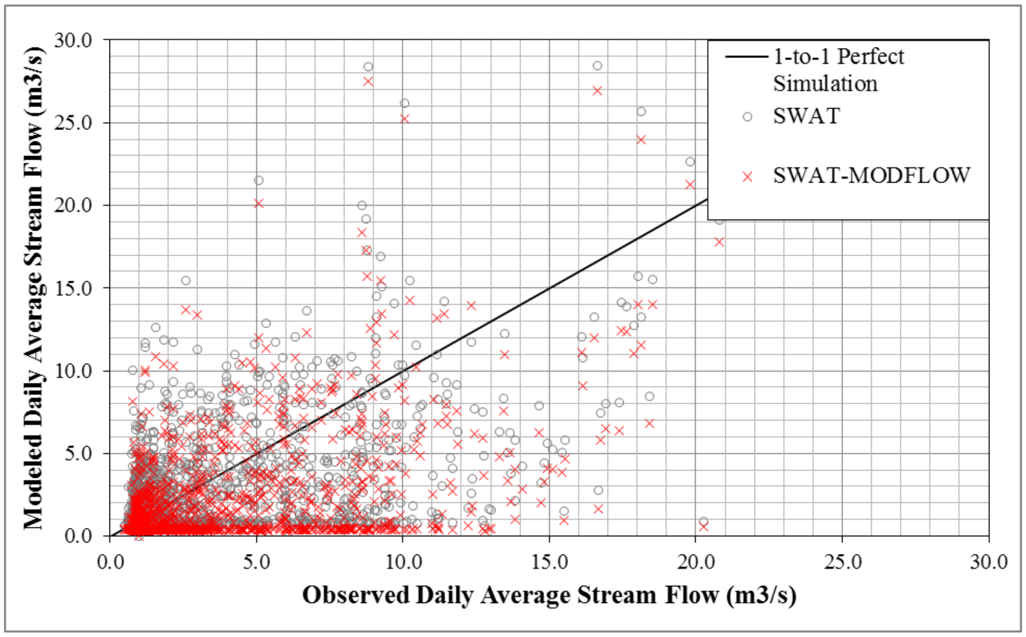


Figure 14: 1-to-1 Comparison of Daily SWAT and SWAT-MODFLOW Results for North Fork of the Sprague River

Table 2: Comparison of Daily SWAT and SWAT-MODFLOW Results to Observed Data for the North Fork of the Sprague River

	SWAT	SWAT-MODFLOW
Nash-Sutcliffe	0.23	0.14
R^2	0.35	0.34

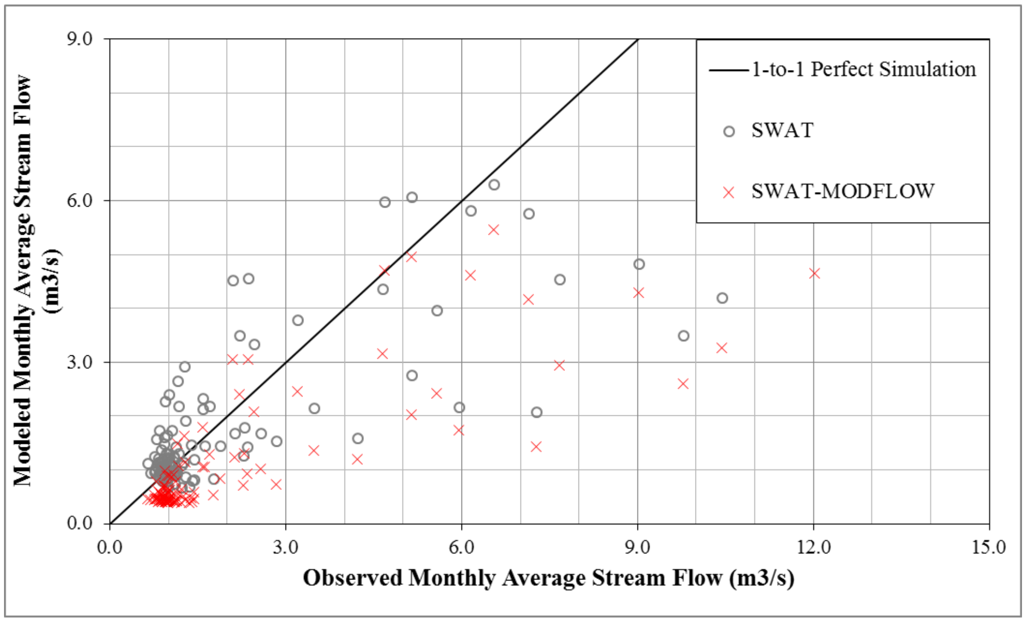


Figure 15: 1-to-1 Comparison of Monthly SWAT and SWAT-MODFLOW Results for North Fork of the Sprague River

Table 3: Comparison of Monthly SWAT and SWAT-MODFLOW Results to Observed Data for the North Fork of the Sprague River

	SWAT	SWAT-MODFLOW
Nash-Sutcliffe	0.57	0.38
R^2	0.60	0.66

To better examine SWAT-MODFLOW's impacts on base-flow representation, the Log10 of both daily and monthly data was taken and compared with observed data in a 1-to-1 plot as shown in Figure 16 and Figure 17, respectively. For both the daily and monthly Log10 comparisons the correlation between observed data and SWAT-MODFLOW simulations appears stronger than correlations between observed data and SWAT. However, the SWAT results are much closer to the 1-to-1 perfect simulation than SWAT-MODFLOW. Some of this error in SWAT-MODFLOW's simulation could potentially be corrected after calibration of the coupled model. The statistical summary for both the daily and monthly comparisons with observed flow are shown in Table 4 and Table 5, respectively. In both daily and monthly cases there is a large increase in R^2 although there is a decrease in the NS of the Log10 data from SWAT (daily R^2 = 0.33, NS = 0.13; monthly R^2 = 0.63, NS = 0.63) to SWAT-MODFLOW (daily R^2 = 0.37, NS = -1.43; monthly R^2 = 0.74, NS = -0.29).

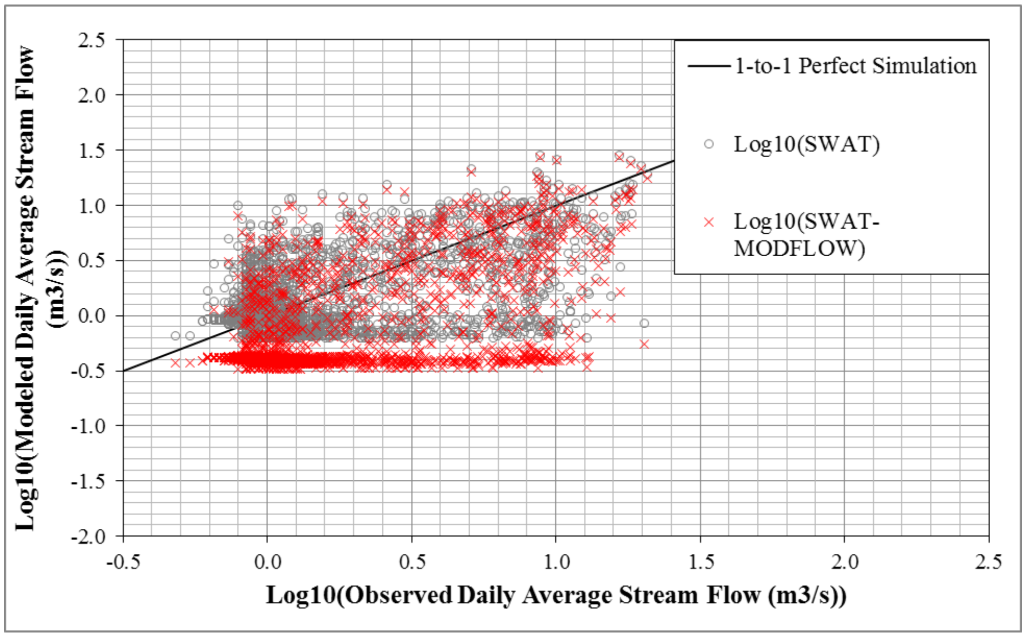


Figure 16: 1-to-1 Comparison of Log-10 of Daily SWAT and SWAT-MODFLOW Results for North Fork of the Sprague River

Table 4: Comparison of Log-10 of Daily SWAT and SWAT-MODFLOW Results to Log-10 of Observed Data for the North Fork of the Sprague River

	SWAT	SWAT-MODFLOW
Nash-Sutcliffe	0.13	-1.43
R^2	0.33	0.37

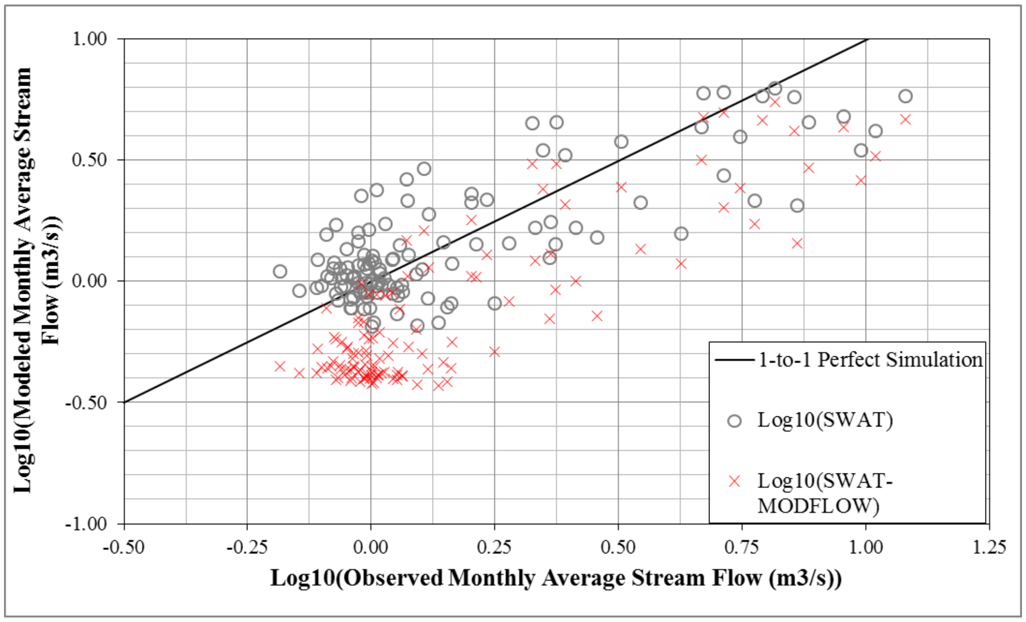


Figure 17: 1-to-1 Comparison of Log-10 of Monthly SWAT and SWAT-MODFLOW Results for North Fork of the Sprague River

Table 5: Comparison of Log-10 of Monthly SWAT and SWAT-MODFLOW Results to Log-10 of Observed Data for the North Fork of the Sprague River

	SWAT	SWAT-MODFLOW
Nash-Sutcliffe	0.63	-0.29
R2	0.63	0.73

Figure 18, which contains a flow duration curve of daily stream flow, illustrates the differences in the stream flow simulation by SWAT and SWAT-MODFLOW. A flow duration curve is a graph of statistically ranked flow data based on its exceedence probability. For example, flows that are exceeded 43% of the time have a value of 43 on the x-axis. As depicted in Figure 18, the SWAT model flow duration curve is close to the shape and magnitude of the observed flows. The SWAT-MODFLOW model, however, under-estimates the low flows of this particular sub-basin by about half a cubic meter per second. Again, it is likely that this performance can be improved by calibrating the coupled SWAT-MODFLOW model.

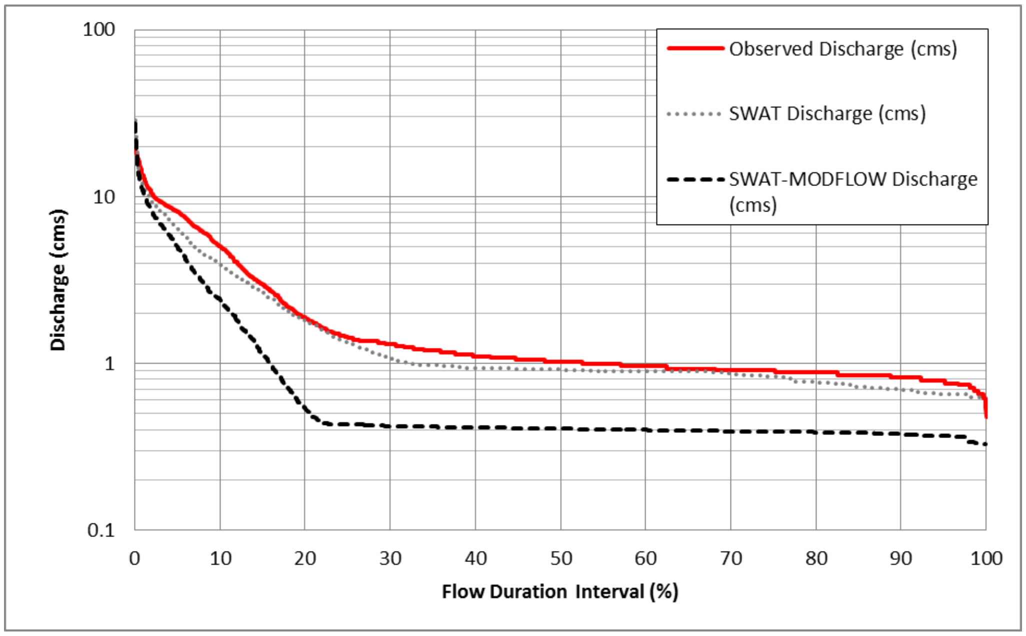


Figure 18: Flow Duration Curve Comparison for North Fork of the Sprague River Stream Flow

A statistical analysis of the monthly stream flow and residual errors between the SWAT and SWAT-MODFLOW models and observed data are summarized in Figure 19. In the monthly error analysis, the SWAT model had a smaller magnitude of error than the SWAT-MODFLOW model. The monthly graphs of the SWAT model match the observed data more closely than the original SWAT-MODFLOW model; however, this may be resolved via calibration of the SWAT-MODFLOW model.

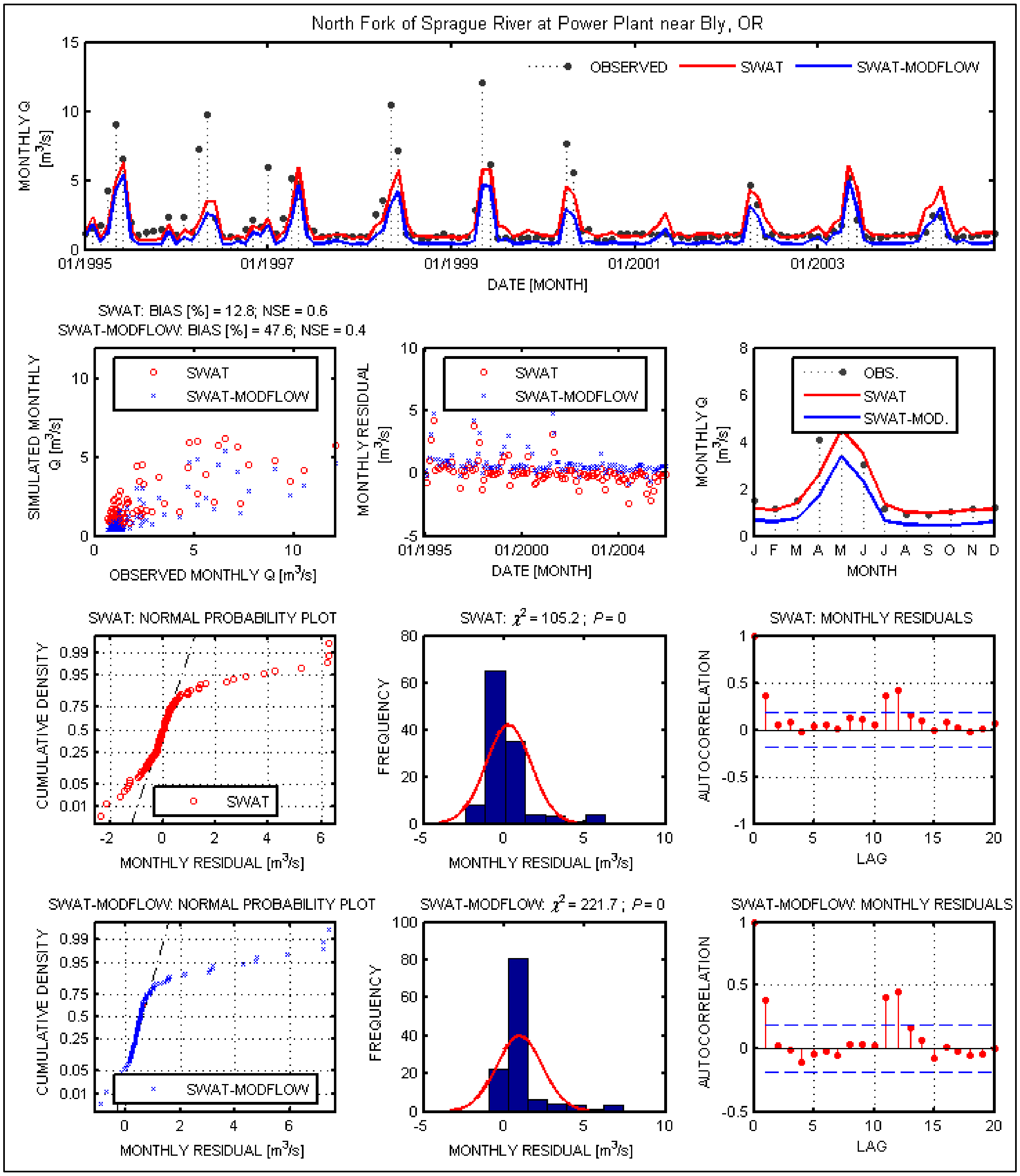


Figure 19: Comparison and Error Statistics of Monthly SWAT and SWAT-MODFLOW Results for North Fork of the Sprague River

2.3.1.2: BASE-FLOW RESULTS

To illustrate the difference in base-flow simulation between the SWAT and coupled SWAT-MODFLOW model, a base-flow separation of the resulting hydrographs was performed using the BFLOW base-flow separation filter developed by Arnold et al. (1995a; Arnold and Allen, 1999a). BFLOW performs three separate base-flow separation passes; usually the first pass is sufficient to capture the base-flow from a hydrograph (Arnold et al. 1995a). A 1-to-1 comparison of the BFLOW's first pass was made of SWAT versus observed and SWAT-MODFLOW versus observed.

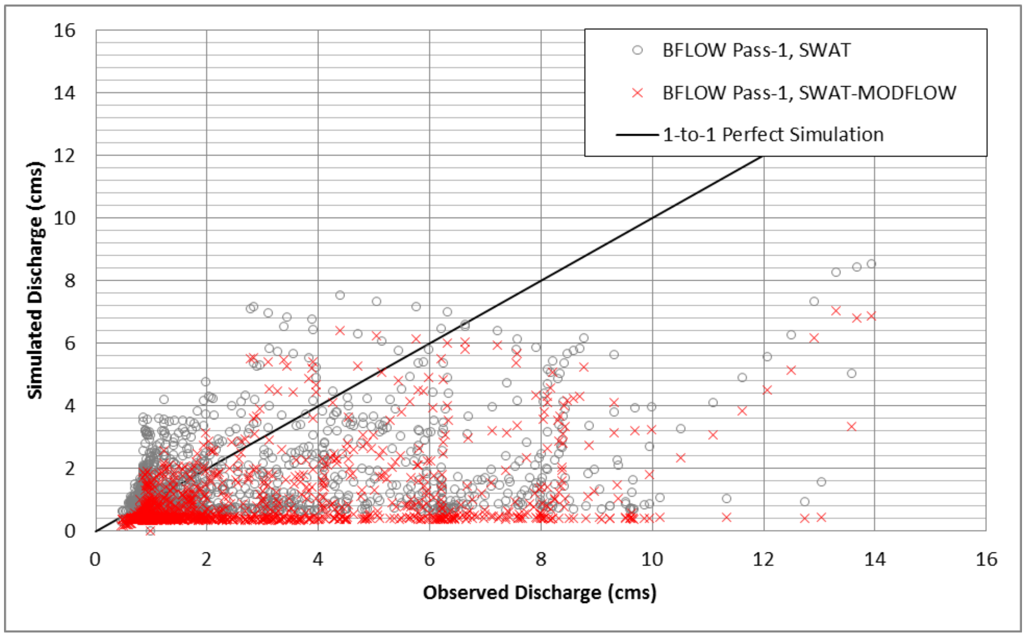


Figure 20: SWAT and SWAT-MODFLOW North Fork of the Sprague River Base-flow Separation Pass 1 Daily Results, 1-to-1 Comparison

2.3.1.3: STREAM SEEPAGE/GAIN RESULTS

Another benefit of the coupled SWAT-MODFLOW model is its ability to simulate spatially variable river-aquifer interactions. In the original SWAT model this process took place at the sub-basin level. With the inclusion of MODFLOW, this interaction now takes place on the grid cell level, which is typically smaller than a sub-basin. The average base-flow discharge to the stream and average seepage to the aquifer are both shown below in Figure 21. The sign of the rate of water in/out of a MODFLOW grid cell determines if it is entering (+) or leaving (-) the aquifer. As seen in Figure 21, the spatial variability of this process is more than the sub-basin level of the SWAT watershed can represent, which is another benefit of a fully distributed groundwater model.

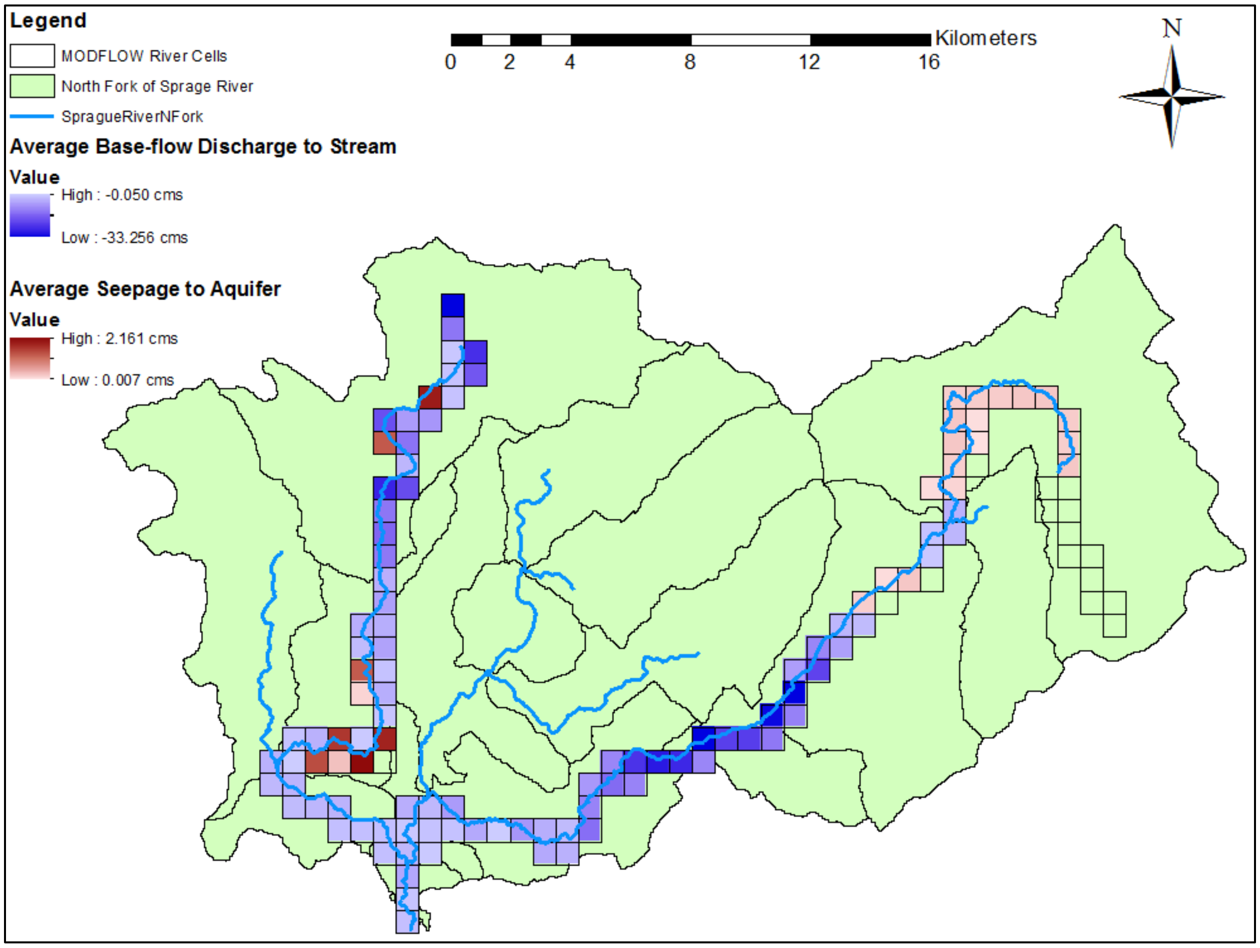


Figure 21: SWAT-MODFLOW North Fork of the Sprague River Stream Seepage/Base-flow Discharge Map

2.3.1.4: WATER TABLE RESULTS

A further comparison of the models addresses the simulation of groundwater height within the basin as simulated by the original MODFLOW model versus the combined SWAT-MODFLOW model for the North Fork of the Sprague River basin. Figure 22 shows the water table heights for the original MODFLOW model after the first stress period of calculations (three months), and illustrates a well-connected continuous aquifer system throughout the North Fork of the Sprague River Basin. Figure 23 illustrates the resulting groundwater table calculated by the SWAT-MODFLOW model after the first stress period. The difference between the MODFLOW model and SWAT-MODFLOW model (MODFLOW result minus SWAT-MODFLOW result) is shown in Figure 24. There are a few locations, primarily the northeast, where the SWAT-MODFLOW groundwater table is different that the MODFLOW model. Normally this might be attributed to the need of a warm-up period of simulation to establish an equilibrium balance because MODFLOW needs this sort of warm-up period. However a warm-up set of initial conditions was previously calculated using a separate steady state analysis performed by Gannett et al. (2012).

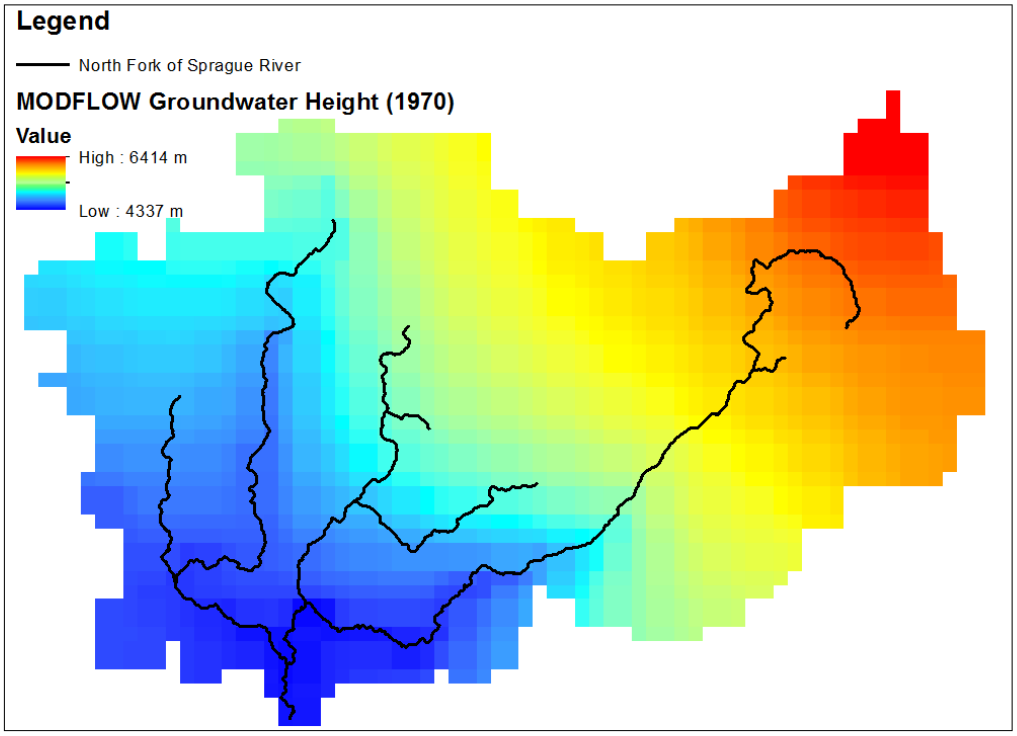

Legend
North Fork of Sprague River
MODFLOW Groundwater Height (1970)
Value
High : 6414 m
Low : 4337 m

Figure 22: Groundwater height at start of MODFLOW simulation for the North Fork of the Sprague River

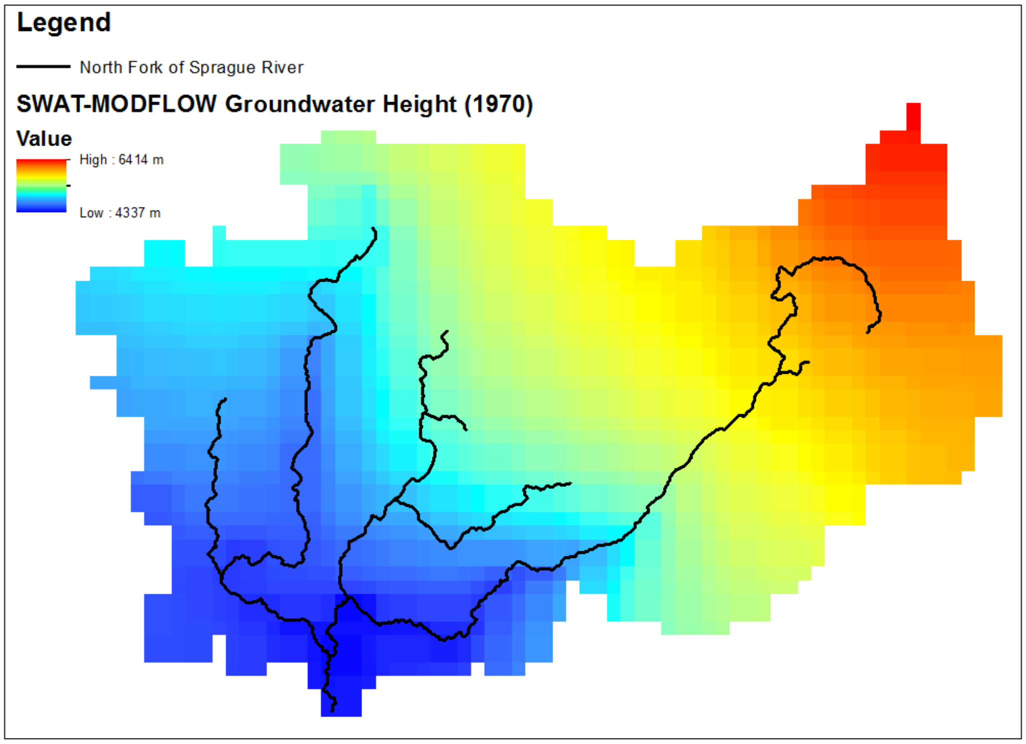

Legend
North Fork of Sprague River
SWAT-MODFLOW Groundwater Height (1970)
Value
High : 6414 m
Low : 4337 m

Figure 23: Groundwater height at start of SWAT-MODFLOW simulation for the North Fork of the Sprague River

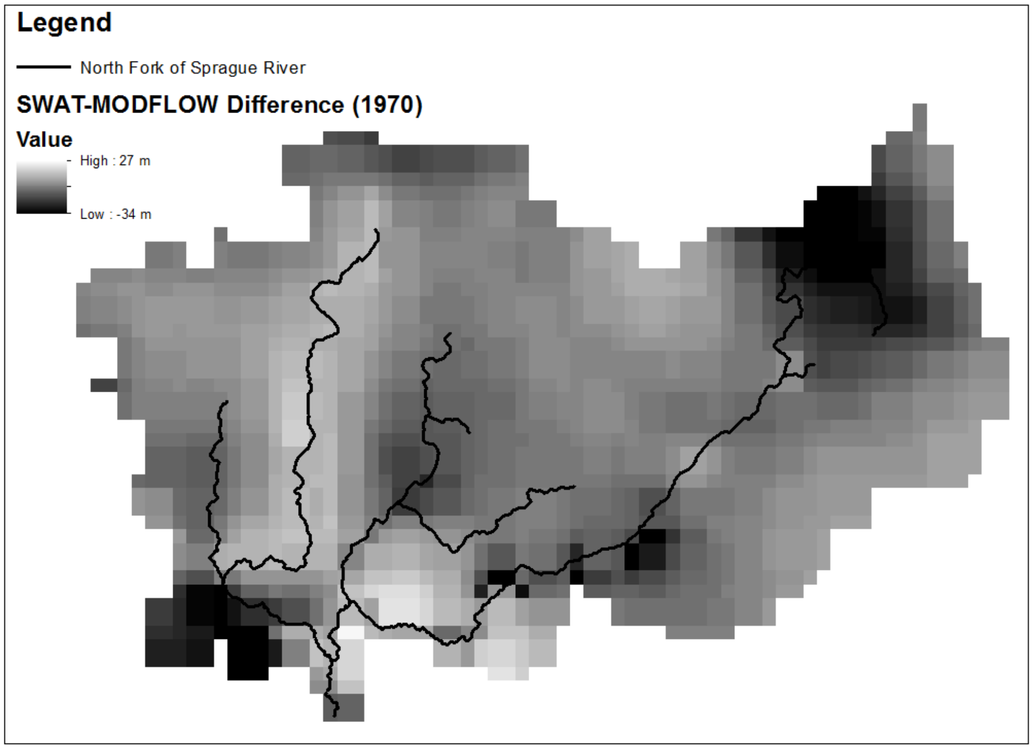


Figure 24: Difference in groundwater height between MODFLOW and SWAT-MODFLOW at start of simulation for the North Fork of the Sprague River

As seen at the end of the simulation results in Figure 25, MODFLOW appears to have minimal change in the aquifer since the start of simulation, some of which is due to the large range of water table values scene within the North Fork of the Sprague River which contains mountains on the east and low river valleys in the south. Figure 26 displays the water table elevations for the coupled SWAT-MODFLOW model at the end of simulation, while the difference between SWAT-MODFLOW and MODFLOW is shown in Figure 27 (MODFLOW result minus SWAT-MODFLOW result). The major difference between the MODFLOW and SWAT-MODFLOW is again located in the northeastern region of the basin where the highest groundwater and ground surface elevations are located.

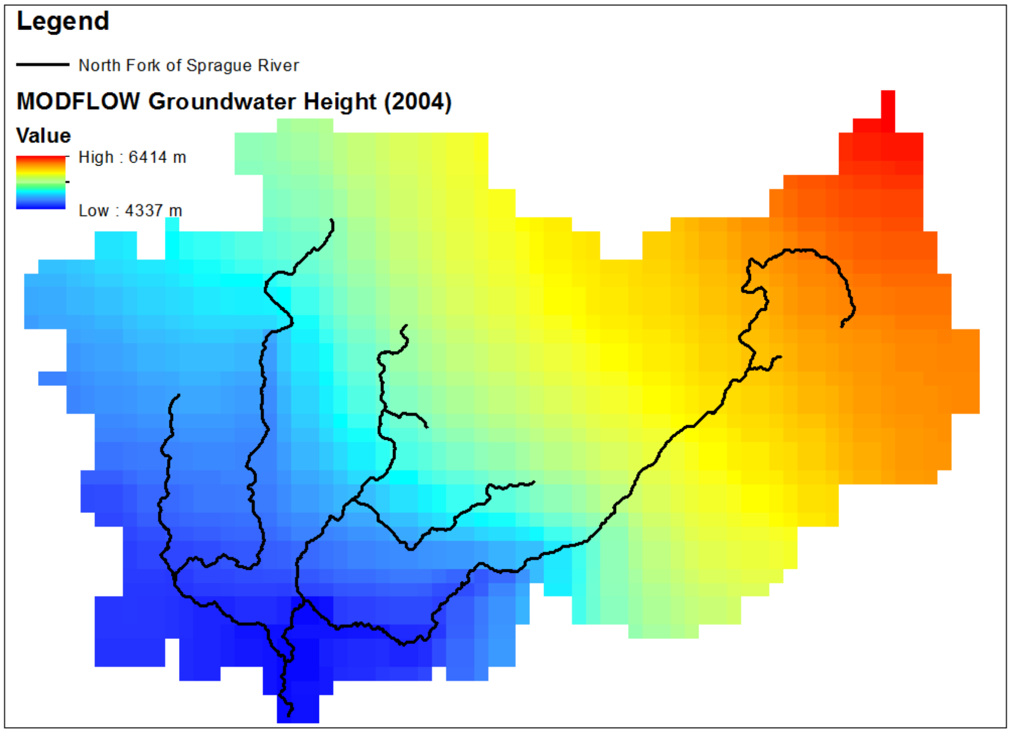

Legend
North Fork of Sprague River
MODFLOW Groundwater Height (2004)
Value
High : 6414 m
Low : 4337 m

Figure 25: Groundwater height at end of MODFLOW simulation for the North Fork of the Sprague River

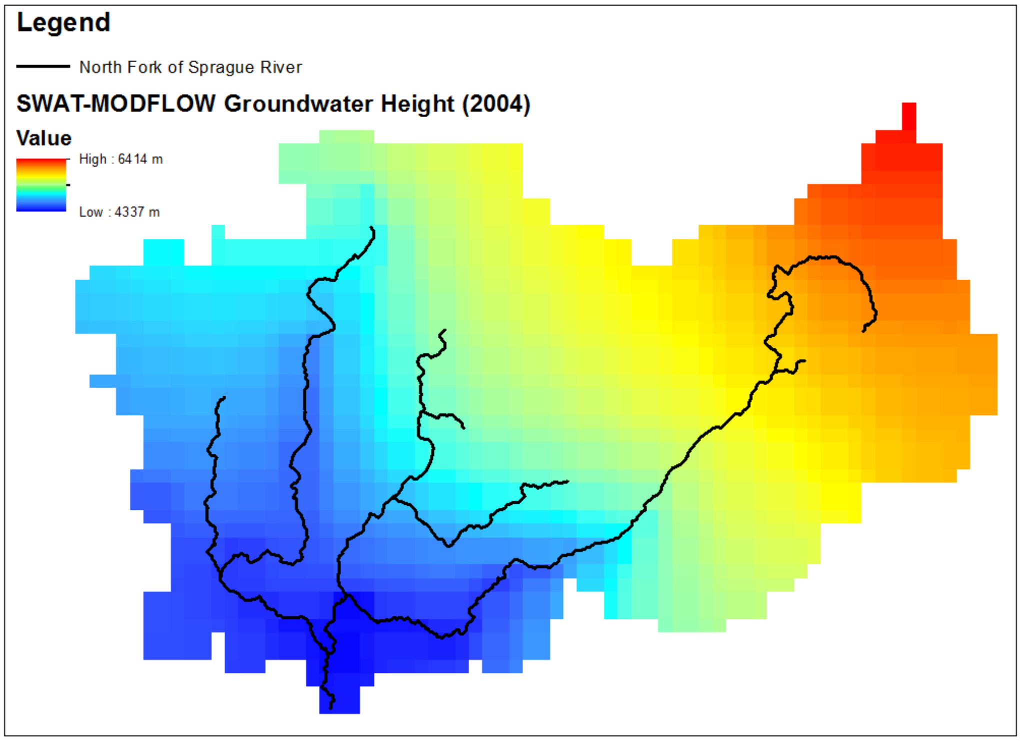

Legend
North Fork of Sprague River
SWAT-MODFLOW Groundwater Height (2004)
Value
High : 6414 m
Low : 4337 m

Figure 26: Groundwater height at end of SWAT-MODFLOW simulation for the North Fork of the Sprague River

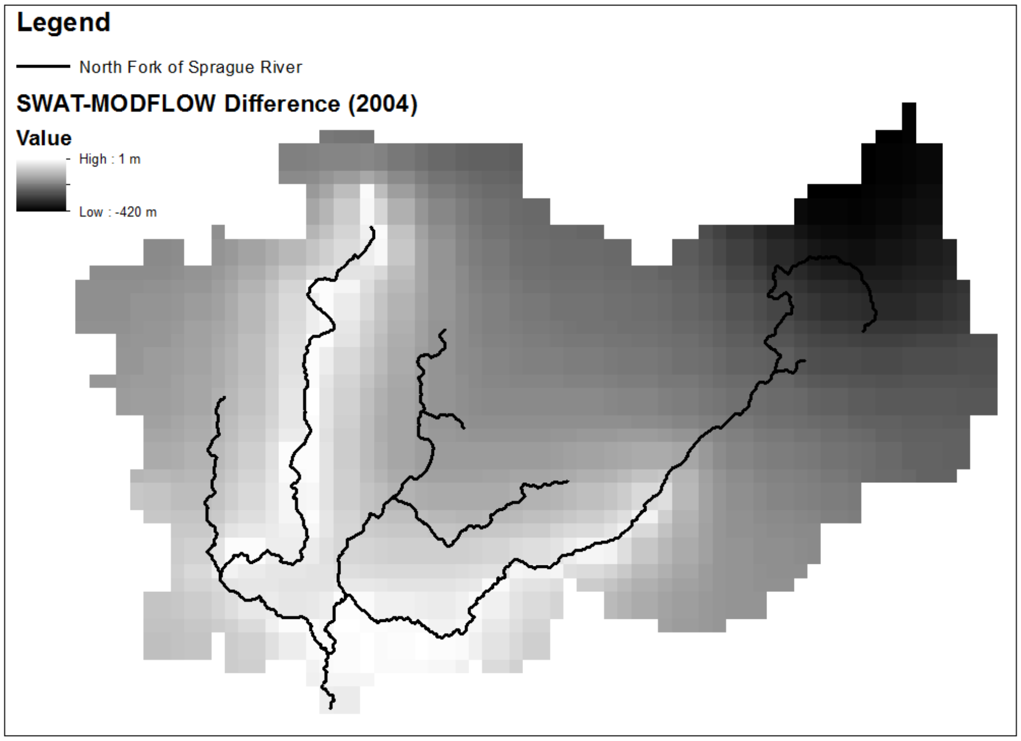


Figure 27: Difference in groundwater height between MODFLOW and SWAT-MODFLOW at end of simulation for the North Fork of the Sprague River

The depressed water table heights at this high elevation region of the basin are due mainly to a smaller volume of recharge entering the aquifer in this location. The difference in recharge is caused by a disagreement between the original MODFLOW model's recharge values, which were determined by the Precipitation-Runoff Modeling System (PRMS) (Gannett et al. 2012), and SWAT's soil percolation values which replace MODFLOW's recharge values in the coupling of the two models within this particular region. The differences in recharge were, however, were confirmed to not be due to the partial spatial coverage of the MODFLOW model by the SWAT model which occurs at this edge of the SWAT model. The generalized spatial linkage of SWAT and MODFLOW in this coupling retains an area-based weighting of the original MODFLOW model value of recharge for any portion of a MODFLOW grid cell that is not contributed to by a SWAT HRU (either partial coverage or no coverage). Therefore, a MODFLOW grid cell which intersects with HRUs for 40 percent of its area receives a recharge value that is 40 percent from SWAT soil zone percolation values and 60 percent from the original MODFLOW model value. A possible reason for the discrepency in water table simulation in the northeast portion of the basin is how the soil zone percolation is calculated. The percolation out of SWAT's soil zone is primarily derived from precipitation at the sub-basin scale. In sub-basins, like this northeast region, precipitation processes can be very different at localized high elevations as compared to the valley areas even though both areas are simulated within a single SWAT sub-basin. To address this, SWAT models allow the use of elevation bands to vary certain watershed parameters based on land surface elevation, which were used in the SWAT model by Records (2013). It appears, however, that there is still some disparity between the groundwater heights as influenced by SWAT soil percolation values and the original MODFLOW model using PRMS aquifer recharge values. Finally, the original MODFLOW model cited sparse availability of calibration data for water table heights in the forested upland areas of the basin (Gannet et al., 2012), resulting in a higher margin of error in these locations for water table elevation.

With the exception of the issue in the northeast portion of the model for groundwater elevations, the coupled SWAT-MODFLOW model mostly reflects groundwater levels close to the original Upper Klamath MODFLOW model, indicating that the soil zone percolation values from SWAT in the North Fork of the Sprague River are comparable with the PRMS recharge values used by the MODFLOW. Additionally, the coupled SWAT-MODFLOW model simulated a higher water table than the original MODFLOW model, which is actually an improvement over the original MODFLOW model because, as explained by Gannet et al. (2012), the original MODFLOW model simulated groundwater elevations residuals within the Sprague River basin of roughly 100 to -100 ft, with most of the simulations being lower than observed water table levels. No comparison was made to observation wells because data was unavailable within the catchment area of interest in the North Fork of the Sprague River Basin.

2.3.2: SPRAGUE RIVER

2.3.2.1: STREAM FLOW RESULTS

Upon examination of the SWAT and SWAT-MODFLOW results for the whole Sprague River watershed, the results were almost identical. Both the SWAT and SWAT-MODFLOW stream flow values were equivalent to 2 decimal places at the outlet of the watershed. The comparison against observed data at USGS Station 11501000 is shown below in Figure 13. The SWAT and SWAT-MODFLOW results are so similar that, when plotted, they lie on top of one another. Both models do a good job simulating the majority of stream flows. However, the recession rates of the simulated hydrographs do not match the observed values. Statistical summaries for the fit of the models to daily and monthly observed data are shown in Table 2 and Table 3, respectively. The daily and monthly statistics for SWAT and SWAT-MODFLOW are equal due to the similar stream flow outputs. Due to the similarity between the model results, a water balance was also examined for the models. The SWAT model was found to calculate a zero groundwater discharge over the analysis period from 1995-2004. The SWAT-MODFLOW model was found to calculate a groundwater discharge approximately equal to 0.2% of the total stream flow. It is believed that since the simulated groundwater discharge represents such a minor portion of the total stream flow, the SWAT-MODFLOW model yielded similar results to the SWAT model.

An explination for this ‘washing out’ of the groundwater discharge as compared to total stream flow is found by examining the river system in question. The most groundwater influence is found in the North Fork of the Sprague River. However, the largest tributary to the Sprague River is the Sycan River which is approximately twice as large as the North Fork of the Sprague River. The final South Fork of the Sprague River, the last significant tributary, is a surface process dominated tributary like the Sycan. This combination of several large surface-driven streams and a single small groundwater influenced stream effectively overpowers small trends in total stream flow due to base-flow discharge from groundwater.

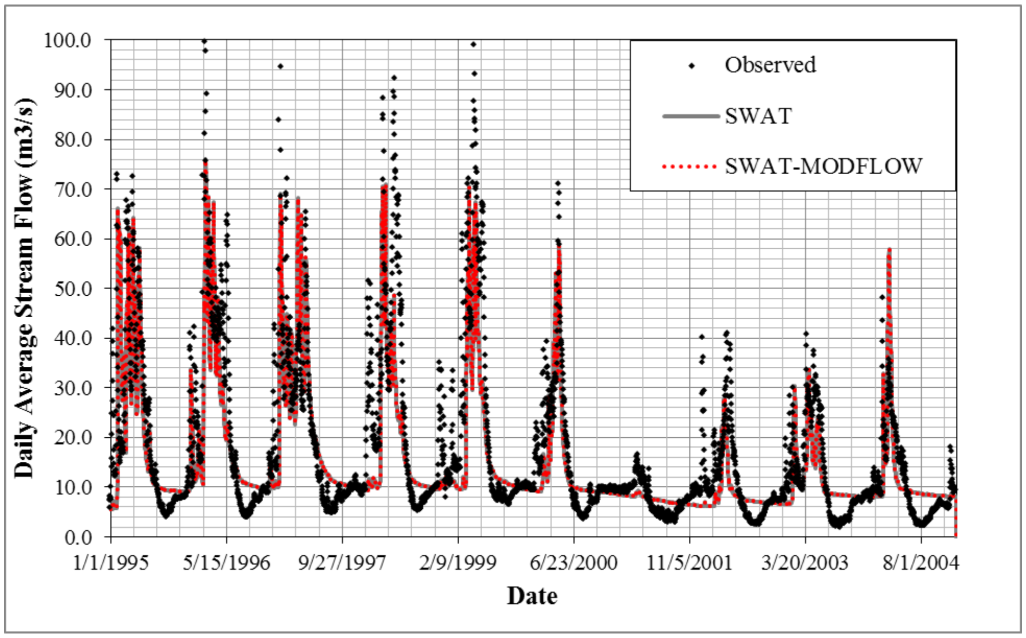


Figure 28: SWAT-MODFLOW Daily Results, No Additional Calibration for the Sprague River, OR

Table 6: Comparison of Daily SWAT and SWAT-MODFLOW Results to Observed Data for Sprague River

	SWAT	SWAT-MODFLOW
Nash-Sutcliffe	0.54	0.54
R^2	0.59	0.59

Table 7: Comparison of Monthly SWAT and SWAT-MODFLOW Results to Observed Data for the Sprague River

	SWAT	SWAT-MODFLOW
Nash-Sutcliffe	0.68	0.68
R^2	0.76	0.76

A further illustration of the similarities in the stream flow simulation by SWAT and SWAT-MODFLOW is shown in Figure 18 , which contains a flow duration curve for total stream flow. Note that this flow duration curve is shaped differently than the curve for the North Fork of the Sprague River which indicates a different series of controls and contributions to flow, primarily a lack of base-flow dominance (large flat curve). Again, the SWAT and SWAT-MODFLOW model results are so similar that they plot on top of one another. This flow duration curve also highlights the inability of either model to capture the observed recession rates in low flow periods (flow durations > 20%). Additionally, both models slightly under predict the higher flows observed in the basin.

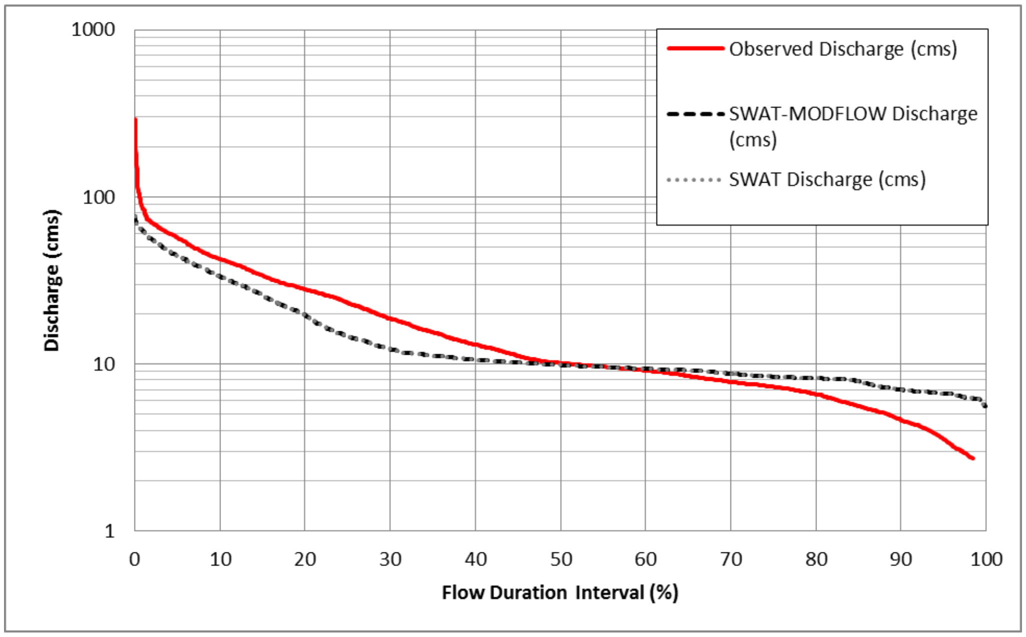


Figure 29: Flow Duration Curve Comparison for the Sprague River Stream Flow

2.3.2.2: STREAM SEEPAGE/GAIN RESULTS

As with the North Fork of the Sprague River analysis, the average base-flow discharge from the aquifer to the stream and seepage from the stream to the aquifer is shown in Figure 30. These results are similar to those from the North Fork analysis, and their implications were discussed previously.

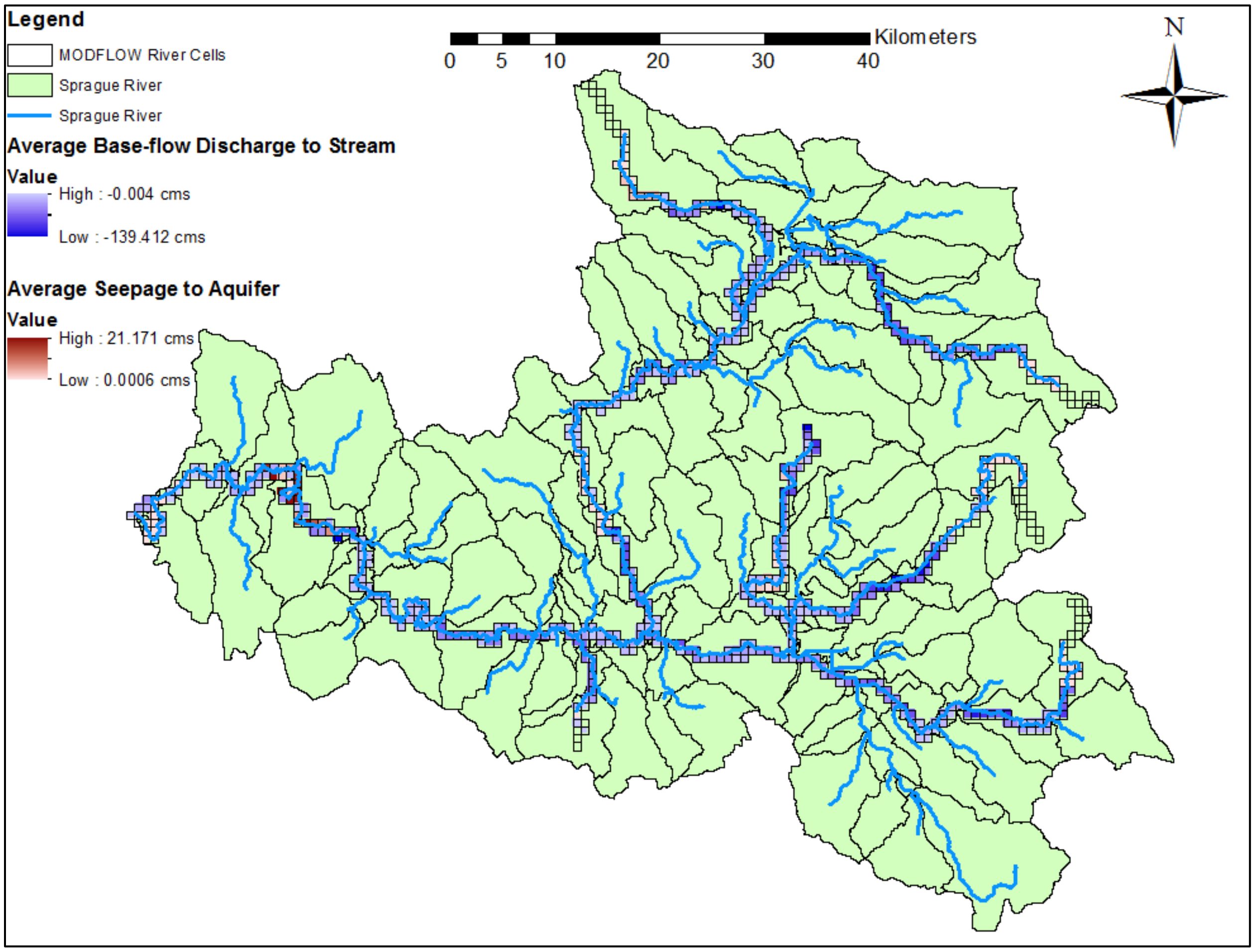


Figure 30: SWAT-MODFLOW Sprague River Stream Seepage/Base-flow Discharge Map

2.3.2.3: WATER TABLE RESULTS

As compared earlier for the North Fork of the Sprague River, the SWAT-MODFLOW results closely resemble those of the original MODFLOW model. Shown in Figure 31 and Figure 32 are the groundwater table heights for the entire Sprague River after the first stress period in 1970 for the MODFLOW and the SWAT-MODFLOW models, respectively. The difference between the values is shown in Figure 33. There are numerous small differences, but the probable cause of these has been discussed in the North Fork’s water table result section. One new difference to emphasize is that at the start of simulation there appears to be a large difference in one of the Sycan River's tributaries towards the top of the map in Figure 33, while the rest of the basin is close to or higher than the original MODFLOW model.

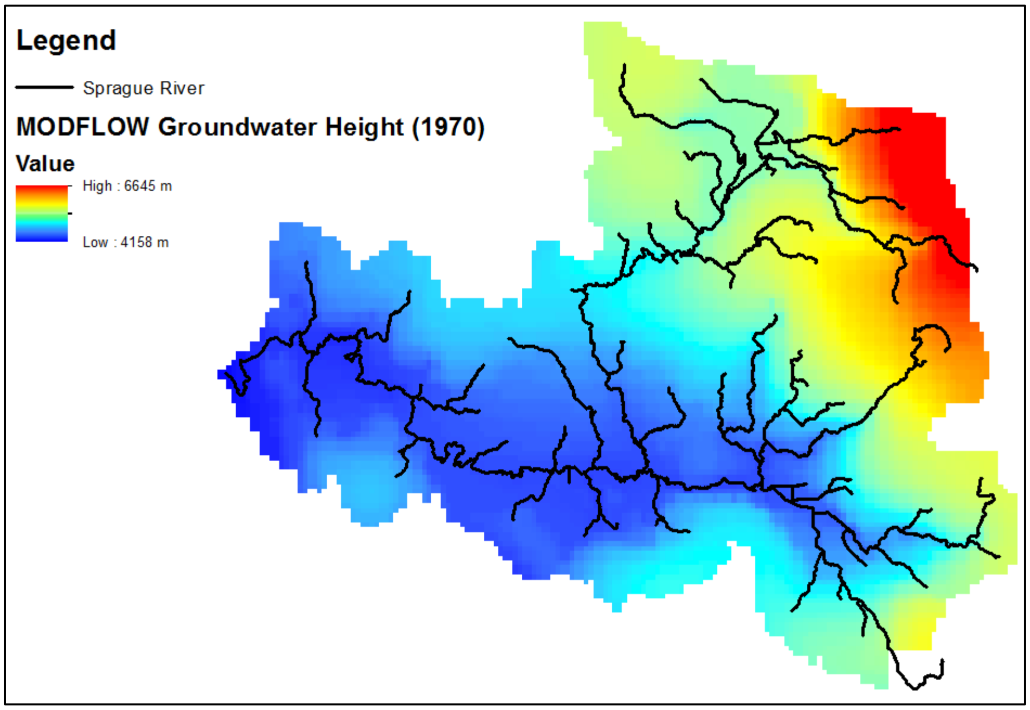


Figure 31: Groundwater height at start of MODFLOW simulation for the Sprague River

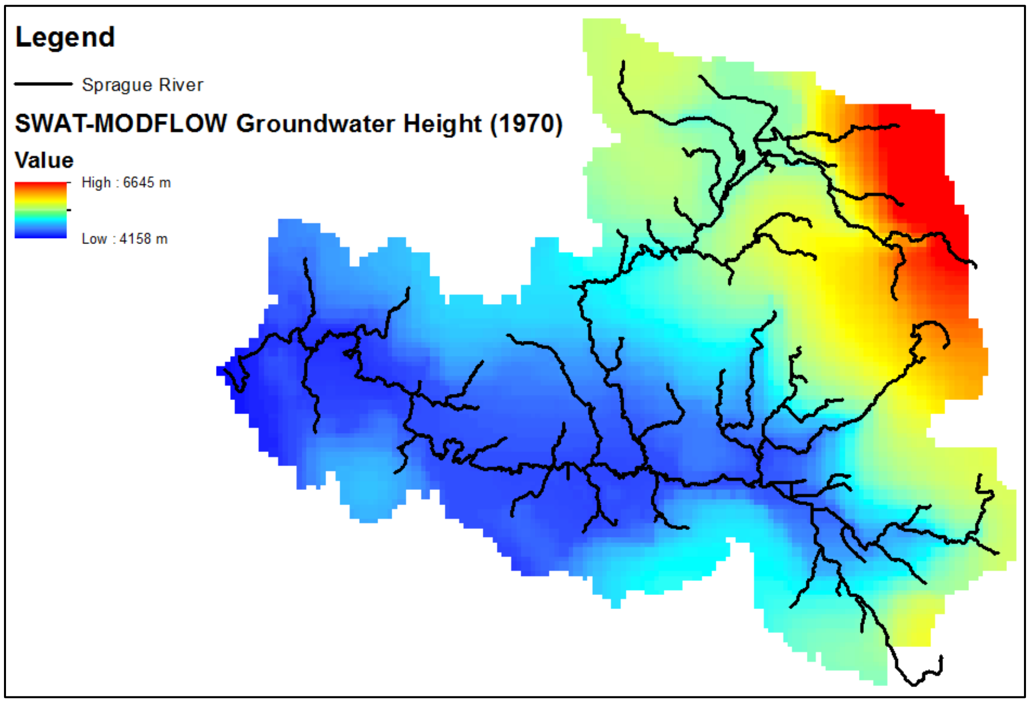


Figure 32: Groundwater height at start of SWAT-MODFLOW simulation for the Sprague River

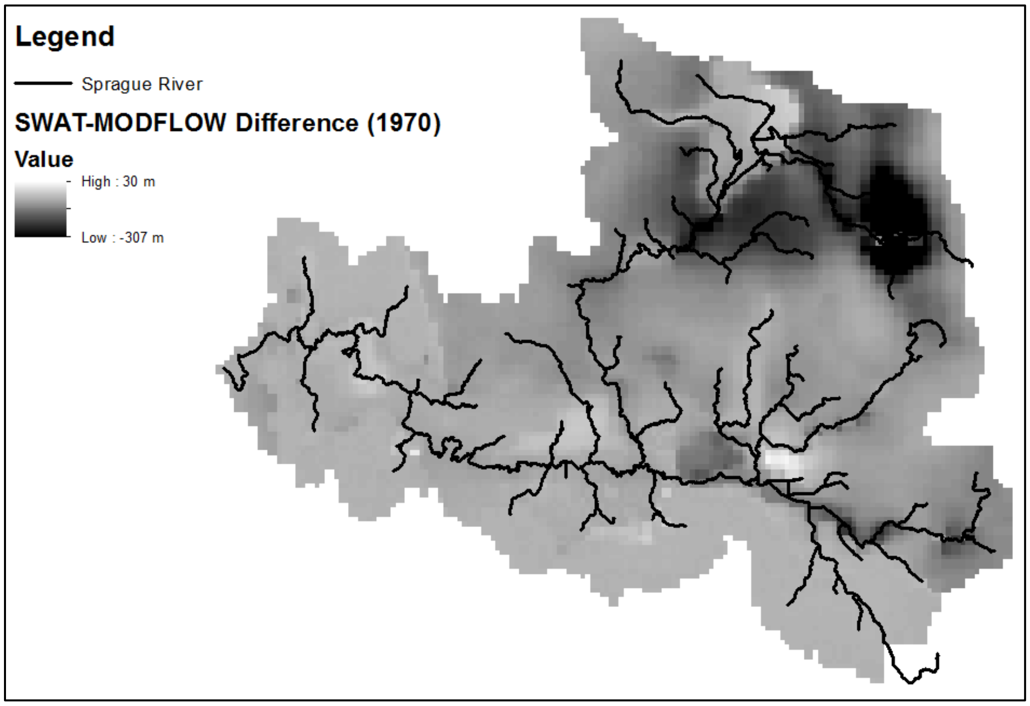


Figure 33: Difference in groundwater height between MODFLOW and SWAT-MODFLOW at start of simulation for the Sprague River

Summarized in Figure 34 and Figure 35 are the groundwater heights at the end of simulation in 2004 for the MODFLOW and SWAT-MODFLOW models, respectively. Again both models simulate a continuous aquifer which patterns itself roughly parallel to the area's land surface. The differences between the MODFLOW model and the SWAT-MODFLOW model are shown in Figure 36. One thing to note on this map is the large area of difference in the northeastern corner of the Sprague River Basin. The likely causes of this difference are summarized in the North Fork water table results section.

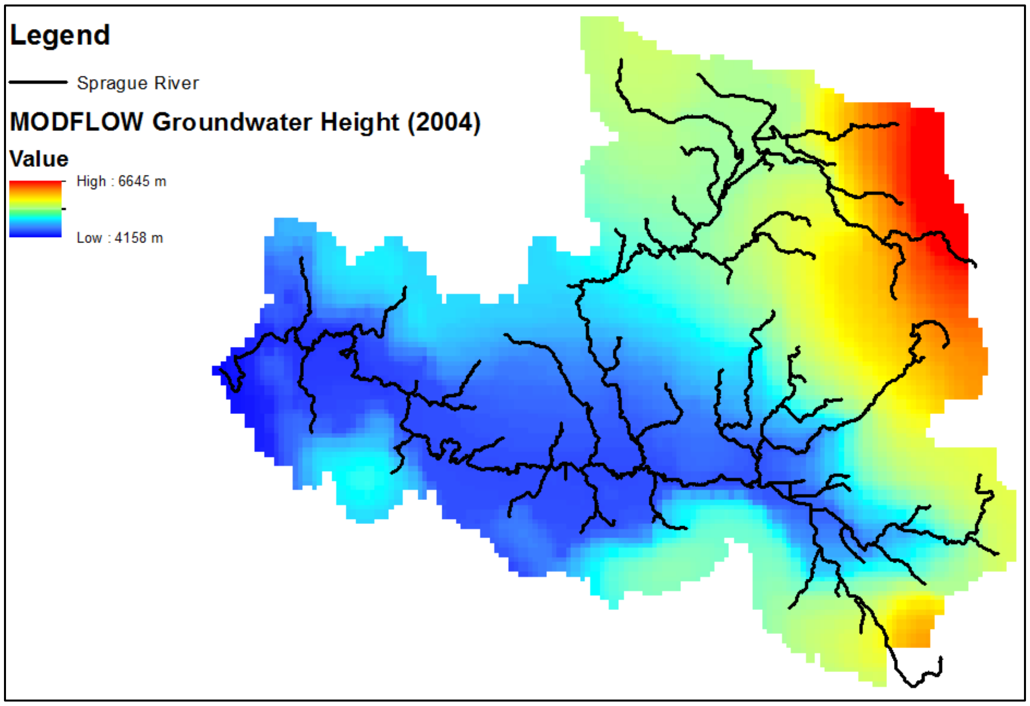


Figure 34: Groundwater height at end of MODFLOW simulation for the Sprague River

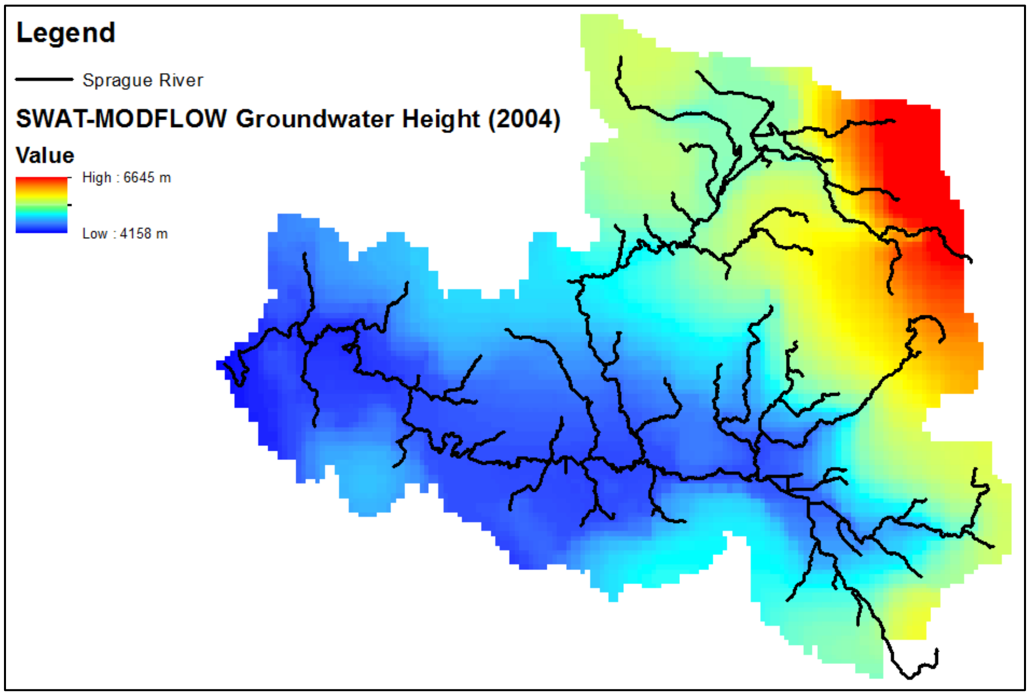


Figure 35: Groundwater height at end of SWAT-MODFLOW simulation for the Sprague River

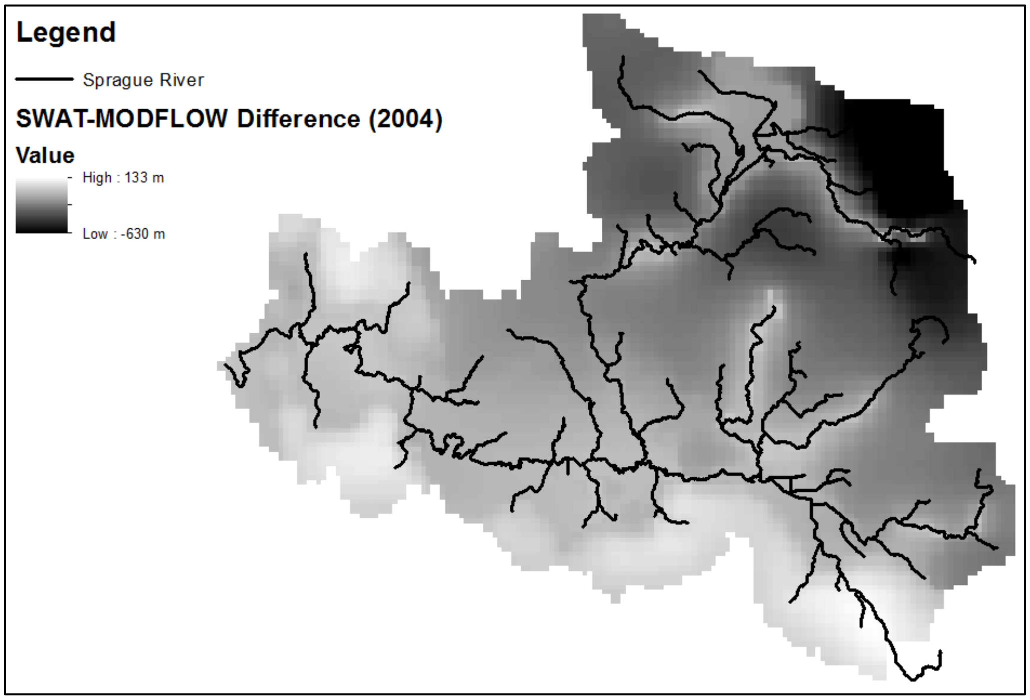


Figure 36: Difference in groundwater height between MODFLOW and SWAT-MODFLOW at end of simulation for the Sprague River

2.3.4: LIMITATIONS AND FUTURE WORK

One consequence of the daily SWAT-MODFLOW coupling is the greater execution time of the model. The original MODFLOW model for the Upper Klamath Basin solves the groundwater flow equations every quarterly stress period. By forcing MODFLOW to run on a daily time step, it increases the number of groundwater flow equation solutions by 90 times the original model. Based on an initial run time of approximately 12 minutes, for an Intel Core 2 Duo 2GHz CPU with 4GB of RAM, this forecasts an anticipated 18 hours to run MODFLOW on a daily time-step for the 35 year Upper Klamath Basin model. However, due to the similarity of much of the watershed on a daily basis combined with the upgrade of the Upper Klamath Basin model from MODFLOW-2000 to MODFLOW-2005-NWT, and Sprague River model from SWAT-2009 rev. 477 to SWAT-2012 rev. 591 resulted in the coupled SWAT-MODFLOW model completing a 35 year analysis in approximately 11 hours. This is does not seem unreasonable compared to the Lower Arkansas River Valley (LARV) MODFLOW-only model documented by Morway et al. (2013) to focuses on the groundwater alluvial aquifer impacts due to heavy irrigated agriculture which takes approximately 9 hours to execute.

A continuation of after a coupled surface-subsurface watershed model would be to add the capacity to simulate nutrient transport between SWAT and MODFLOW. SWAT already has a nutrient model in it however MODFLOW does not. In order to simulate subsurface nutrient transport and interaction a third model would need to be combined to SWAT-MODFLOW like the subsurface chemical/nutrient transport model like MT3DMS or UZF-RT3D. Coupling this subsurface reaction, advection, and dispersion transport model would allow the combined surface-subsurface watershed would simulate not only hydrology but also nutrient, metal, and pesticide transport at the watershed scale. This sort of coupling would assist in tackling the complex watershed chemical problems like the nutrient and wetland issues in the Upper Klamath basin and the nitrate-selenium issues in the Lower Arkansas River Valley.

2.4: CONCLUSIONS

The complex issues of water demand and environmental change in watersheds have driven the desire to simulate and model watersheds and the impacts of various management and climate change scenarios. The pseudo-distributed watershed model SWAT and the finite-difference groundwater model MODFLOW are at the forefront of popular available watershed models. Inabilities of these models to simulate complex groundwater response as well as erosion, plant growth, nutrient cycling, and agricultural management has led to multiple attempts to link the models. However, these models are unavailable and have drawbacks like required identical spatial discretization and monthly coupling. As a result the aforementioned work was undertaken to create a generalized spatial linkage between SWAT and MODFLOW which retains the respective strengths of the two models coupled on a daily time-step. This linkage facilitates the use of existing spatially dis-similar SWAT and MODFLOW models while increasing the overall quality of simulation for a more reliable result than previously possible.

This daily coupling between SWAT and MODFLOW has resulted in realistic and accurate stream flow results for the North Fork of the Sprague River in Oregon. Additionally, groundwater representation within SWAT was increased by the inclusion of MODFLOW and accurate groundwater table elevations were simulated as a result. However, at the high elevation edge of the watershed a reduced volume of recharge to the aquifer caused lower water table elevations relative to the original MODFLOW model for the Upper Klamath Basin. The new SWAT-MODFLOW coupling resulted in a more accurate representation of both the frequency and magnitude of streams flows than the original calibrated SWAT model only. This model coupling of the coupled SWAT-MODFLOW model also provided these results without additional calibration. It is likely calibration will improve all the results herein discussed.

CHAPTER 3: SCALABLE AND ACCESSIBLE CLOUD-BASED SERVICES FOR STREAM WATER ANALYSIS

3.1: INTRODUCTION

Stream flow data has become an increasingly important tool for assessing current stream conditions as well as a basis for predicting future conditions and water supply. However, current stream flow analysis software typically focuses on only one aspect of stream flow resulting in the need for many software packages to analyze the multiple aspects of stream flow; floods, droughts, groundwater contribution, frequency-duration, pollutant loading, and more. This requires multiple input files to satisfy each of the software packages for one stream gauge dataset. Additionally, these individual packages are desktop-based software that uses local computer resources rather than taking advantage of some of the recent advancements in cloud-computing technology.

Due to this current style of implementation, one analysis in one software package, there are numerous available flow analysis software packages. Base-flow separation, or the separation of groundwater from surface runoff contribution to total stream flow, has been a particular focus with numerous software packages. The Hydrograph Separation tool (HYSEP) was developed by the USGS to help automate a previously manual hydrograph separation technique (Sloto and Crouse, 1996). Another similar base-flow separation package is the BFLOW multi-pass digital-filter base-flow separation tool developed by Arnold et al. (1995a; Arnold and Allen, 1999a). In addition to base-flow software, Flynn et al. (2006) have developed a package, PeakFQ, which automates the U.S. flood analysis, Bulletin 17B (IACWD, 1982), by fitting a regression to available flood. There is also the load estimator tool, LOADEST (Runkel et al., 2004), which estimates in-stream pollutant loads with regression-based interpolation between observed pollutant loads and the Sanitary Sewer Overflow Analysis and Planning tool SSOAP even assists with simulating the impacts of sanitary sewer over flow (Vallabhaneni et al., 2012).

There has, however, been a recent shift in the implementation style of these packages. A number of new software analyses area being ported to the web with internet-based graphical user interfaces to assist with interacting with the analysis. One example of a base-flow separation tool is the Web-based Hydrograph Analysis Tool, WHAT, developed by Lim et al. (2005). Govindaraju et al. (2009) took it a step further and conceptualized an entire outline for the necessary cyberinfrastructure to support an end-to-end approach to environmental modeling. Another recent web implementation of a flow/water quality analysis tool came in the form of the SPARROW-DSS tool to assist with sediment and nutrient loadings to river basins in the U.S. (Booth et al., 2011). A more complex and necessary step than making a tool available on the web is implementing it in a scalable fashion. One example of a scalable infrastructure for web tools is a cloud-based environment to take advantage of lumped server resources rather than local computer resources. There are many benefits to cloud-infrastructure including greater scalability and the ability to process more than one analysis at a time due to multiple virtual machines. One flow analysis package that has taken advantage of these benefits is ParFlow, a parallel surface-subsurface watershed model, which was implemented in a cloud environment by Burger et al. (2012).

To utilize recent advancements in computer technology, the focus of this additional work is to develop and demonstrate a scalable cloud-computing web-tool that facilitates access and analysis of stream flow data. The specific objectives are to 1) unify the various stream flow analysis topics into a single tool; 2) to assist in the access to data and inputs for current flow analysis methods; 3) to examine the scalability benefits of a cloud-based flow analysis tool.

3.2: METHODS

3.2.1: COMPREHENSIVE FLOW ANALYSIS (CFA) OVERVIEW

The primary focus of this research is to integrate the various aspects of flow analysis and implement it on a scalable cloud-computing web-tool to facilitate access to the tool. The Comprehensive Flow Analysis (CFA) tool was developed by creating and integrating multiple available stream flow analysis methods used for the various aspects of rivers; floods, drought, water quality, pollutant loadings, base-flow contribution from groundwater. In order to increase the access and scalability of the analyses within CFA, CFA was built into the Cloud Services Innovation Platform (CSIP). CSIP is an environmental modeling service which facilitates a scalable cloud infrastructure for analysis execution. The analysis capabilities of CFA are demonstrated by a downstream analysis of nutrient loading on the Cache La Poudre River in Colorado while the scalability of CFA methods are demonstrated for test cases containing all available flow data from 1000 existing USGS stream monitoring sites across the U.S.

3.2.2: CYBERINFRASTRUCTURE

A key importance of this new flow analysis tool is its cyberinfrastructure. CFA was designed for access to the tool through the Environmental Risk Assessment and Management System (eRAMS) website. eRAMS was developed at Colorado State University to facilitate geospatial manipulation of data for use with environmental modeling. eRAMS is built on a web-based geospatial analyst, similar to ArcGIS, allowing data manipulation, environmental modeling and results analysis, and the sharing of geospatial information. The map-based structure of eRAMS allows unique location based searches and access to stream flow monitoring sites and their relevant information like drainage area, elevation, and watershed/sub-basin location. Included on eRAMS is a base layer of all of the stream flow monitoring locations in the USGS' National Water Information System (NWIS) and U.S.EPA's STORET/WQX databases. This point layer acts as the first set of inputs for the CFA tool and its analyses allowing the eRAMS CFA interface to pre-process and auto-populate many of the necessary inputs simplifying the overall process of running a flow analysis with CFA. Appendix V, Figure 47 contains the eRAMS GUI and Figure 48, Figure 49, Figure 50, Figure 51, Figure 52, Figure 53, and Figure 54 show example CFA analyses reports for a station on the Cache La Poudre River, CO.

Additionally, included in the CFA tool is an automatic data extraction for stream flow and water quality information from the USGS' NWIS and EPA's STORET/WQX databases. The dynamic access to these databases allows the CFA tool to retrieve the most recent flow and water quality data for stream locations and combine it with local sampling data that a user may have for a quicker analysis with minimal input file data manipulation. The web access to the CFA tool on eRAMS is made possible by the development and inclusion of the CFA tool into the Cloud Services Innovation Platform (CSIP). Simply put, eRAMS is used as an input preprocessor and interface to the CFA tool while the actual analysis of CFA is carried out through a request to CSIP, as illustrated below in Figure 37.

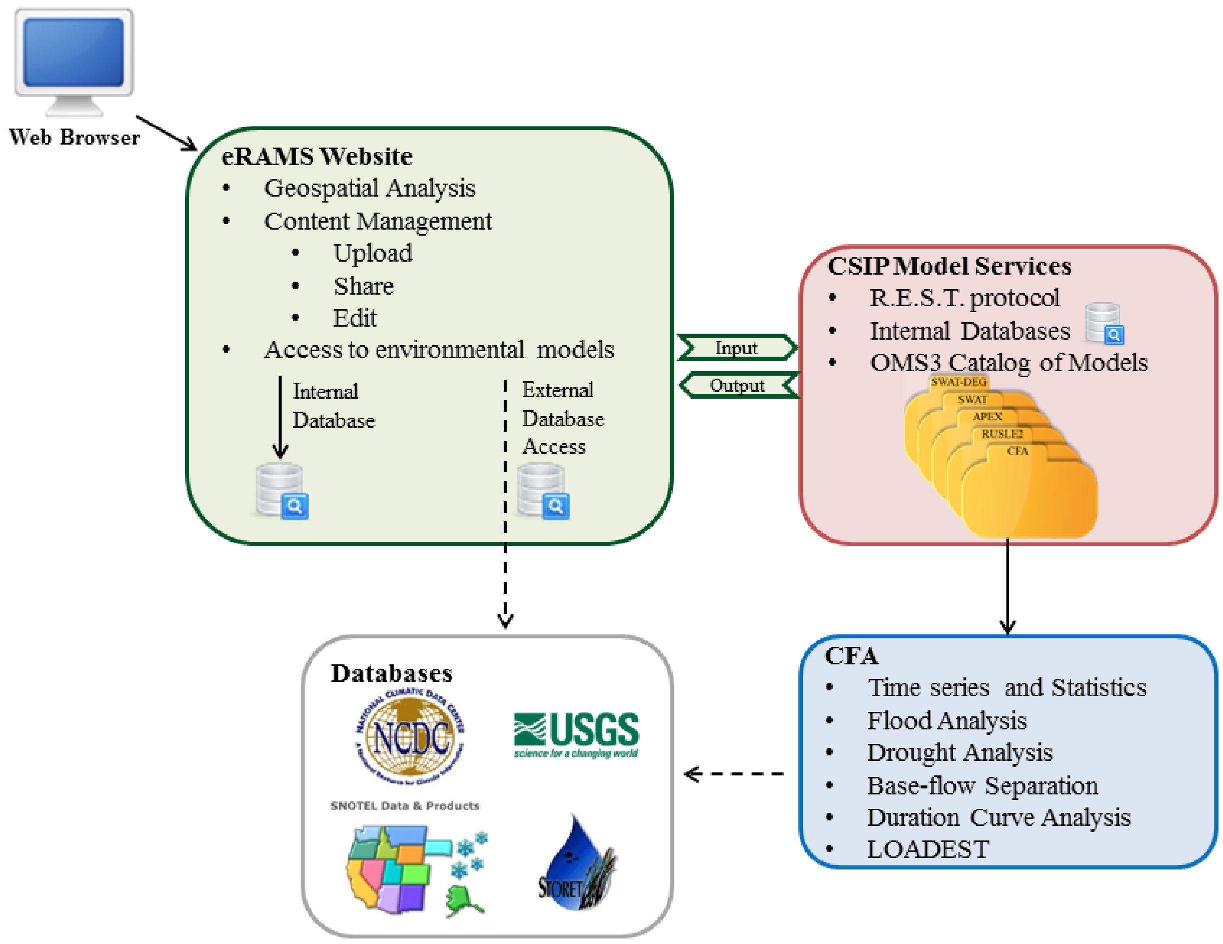


Figure 37: CFA's Interaction with eRAMS, CSIP, and External Databases

3.2.3: SCALABILITY

In order to provide a more scalable infrastructure for the flow analysis tool, CFA was incorporated into CSIP. CSIP deploys its modeling engine using Eucalyptus Infrastructure-as-a-Service Cloud (IaaS) virtual machines (VMs) (Lloyd et al., 2012). CSIP uses Eucalyptus for its ability to provide an elastic modeling platform to manage cloud infrastructure. Within CSIP, Eucalyptus manages the launching, destroying, and modifications to VMs to provide a scalable infrastructure for the CFA tool.

3.2.4: ACCESSIBILITY

As mentioned earlier, the main access to the CFA analyses was incorporated into eRAMS through the development of a graphical user interface (GUI). Additionally, eRAMS provides access to a number of the necessary inputs for CFA as base-layers to the interface. The station name and ID, used for data retrieval within the tool, are provided by the aforementioned base layer of flow stations for USGS and U.S. EPA.

Special data and information for some of the analyses in CFA are also provided by eRAMS as background inputs to the GUI. An example of this additional input is the flood analysis in CFA. CFA's flood analysis performs an automated Bulletin 17B Log-Pearson Type-III distribution regression on available annual flood data. The Bulletin 17B is current the standard practice for analyzing floods on gauged U.S. rivers and streams (WRC-HC, 1967; IACWD, 1982). A complication of the flood analysis recommendations by the Inter-Agency Committee on Water Data (1982) is the requirement of a regionalized flood skewness coefficient in the analysis. A national scale resolution map is provided with the regionalized flood skewness coefficients is provided in the Bulletin 17B documentation, Figure 38, but since it's conception there have been a number of state-scale improvements to this map by state agencies (e.g. Parrett and Johnson, 2004; Soong et al., 2004; Cooper, 2005; Atkins et al., 2009; Olson, 2009; Pomeroy and Timpson, 2010). To simplify the access to the regionalized flood skewness values, a literature review was undertaken to find all the available agency report maps and station skewness coefficient values in addition to the map provided by IACWD (1982). Each state map was then digitized and merged into the IACWD (1982) map to provide a nationwide unified base-layer map used by eRAMS for preprocessing this input for CFA's flood analysis with higher spatial resolution than the original national map where information is available.

Preprocessing this information allows the eRAMS GUI for CFA to automatically extract the regionalized flood skewness coefficient based on the selected station's location.

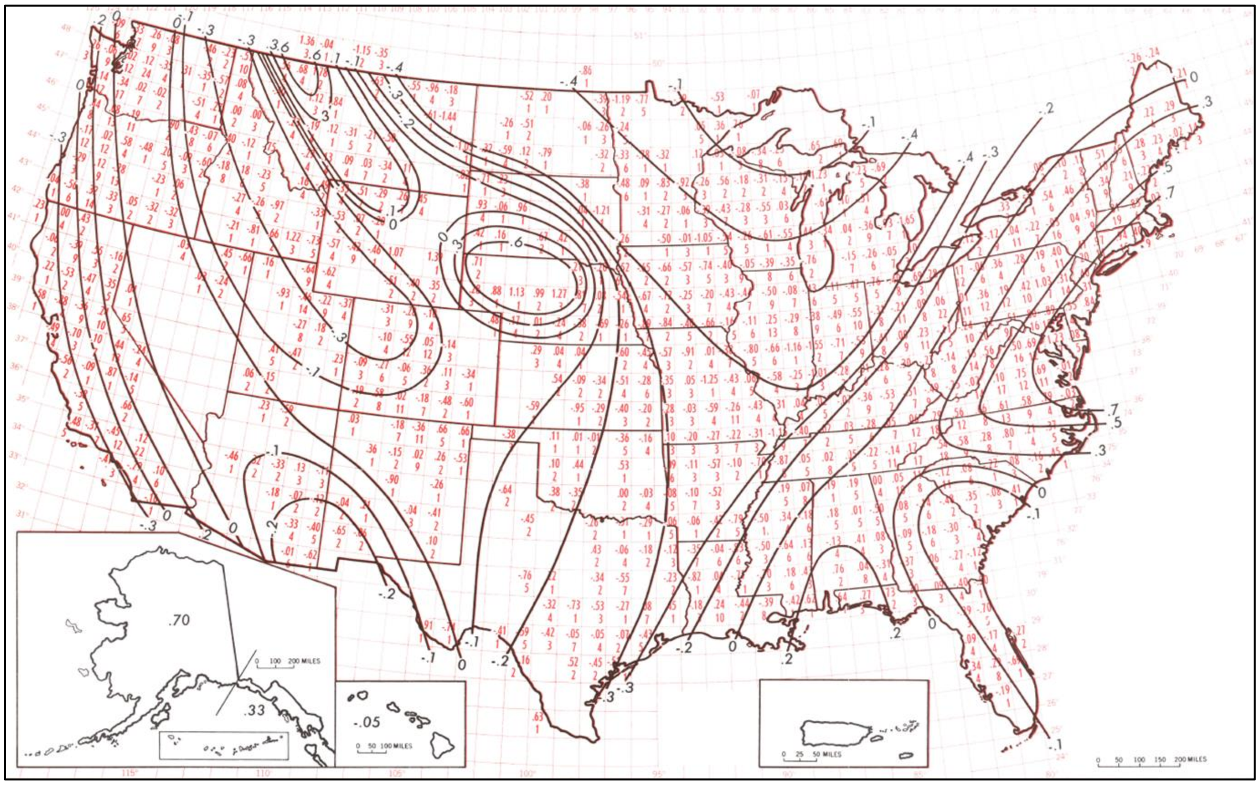


Figure 38: Regionalized Flood Skewness Coefficient Map, Adapted from IACWD, 1982

3.2.5: CFA ANALYSES

The simplest analysis method included in CFA is a time series graphing and statics tool which summarizes available flow data or water quality data, an example of the output for this analysis is shown in Appendix V, Figure 48. As with all the analyses of CFA, the time series analysis is capable of querying the USGS' NWIS and U.S. EPA's STORET/WQX databases for available information based on the provided inputs. A subsequent, and more complex, method in CFA is the flood analysis which, aforementioned, performs an automated Log-Pearson Type-III regression on available flood data based on the provided regionalized flood skewness coefficient (IACWD, 1982). An example of the output for this analysis is shown in Appendix V, Figure 49.

An opposite but equally important aspect of stream flows is the consideration and analysis of droughts from the available stream flow record. Therefore, CFA includes an automated drought analysis method based on the work by Salas et al. (2005). The main component of the drought analysis fits a regression, an autoregressive (AR) or autoregressive-moving average (ARMA), model to annual stream flow data and uses the statistical properties of the original dataset to simulate a much larger dataset which includes rarer long-length, large-deficit 'droughts'. The projected dataset is subsequently compared against a provided long term average drought limit to determine the length, severity, and average recurrence interval of these new 'droughts.' The resulting recurrence interval, severity, and length of each historic drought and projected drought are then summarized graphically. An example of the output for this analysis is shown in Appendix V, Figure 50, Figure 51, and Figure 52 broken into multiple figures due to the size of the output.

Another approach to the analysis of stream flow data is the application of duration curves to graph statistically ranked flow data based on its occurrence. CFA contains two methods that use a duration curve approach, the Flow Duration Curve tool (FDC) and the Load Duration Curve tool (LDC). FDC uses a Weibull plotting position rank to graph daily average stream flows on a scale of percent exceedence allowing a quick visualization of the various flows a river has undergone, an example of the output for this analysis is shown in Appendix V, Figure 53. The LDC follows the same process and takes the analysis a step further and converts the FDC curve into a daily load curve based on a target water quality standard. Furthermore, it superimposes available sampled water quality concentration data (converted based on daily flow values into daily load values) onto the curve to graphically illustrate water quality concentration changes as flow regime changes. LDCs for nutrient can be used to help identify, based on where water quality observations exceed the target curve, probable pollution sources (Cleland, 2002; Cleland, 2003; Cleland, 2007) as well as a basis for establishing total maximum daily load (TMDL) requirements of a watershed. An example of the output for this analysis is shown in Appendix V, Figure 54.

Two additional flow analysis capacities in CFA are provided by the inclusion of currently available software packages for load estimation and base-flow separation. The water quality load estimation software, LOADEST developed by the USGS (Runkel et al., 2004), is built into CFA to act similar to the other analyses and provides an estimation of constituent water quality stream loads between observed data points, an example of the output for this analysis is shown in Appendix V, Figure 55. CFA also includes the base-flow separation software BFLOW, developed by Arnold et al. (1995a; Arnold and Allen, 1999a). BFLOW is an automated multi-pass digital-filter base-flow separation tool to separate the groundwater contribution from total stream flow, an example of the output for this analysis is shown in Appendix V, Figure 56. A technical manual documenting the process of each of the analyses of the CFA tool is provided in Appendix VI.

3.2.6: STUDY AREA

A demonstration of the analysis capabilities of CFA was applied to the Cache La Poudre River Basin in Colorado, Figure 39. A description of the analysis locations shown in Figure 39 is provided in Table 8.

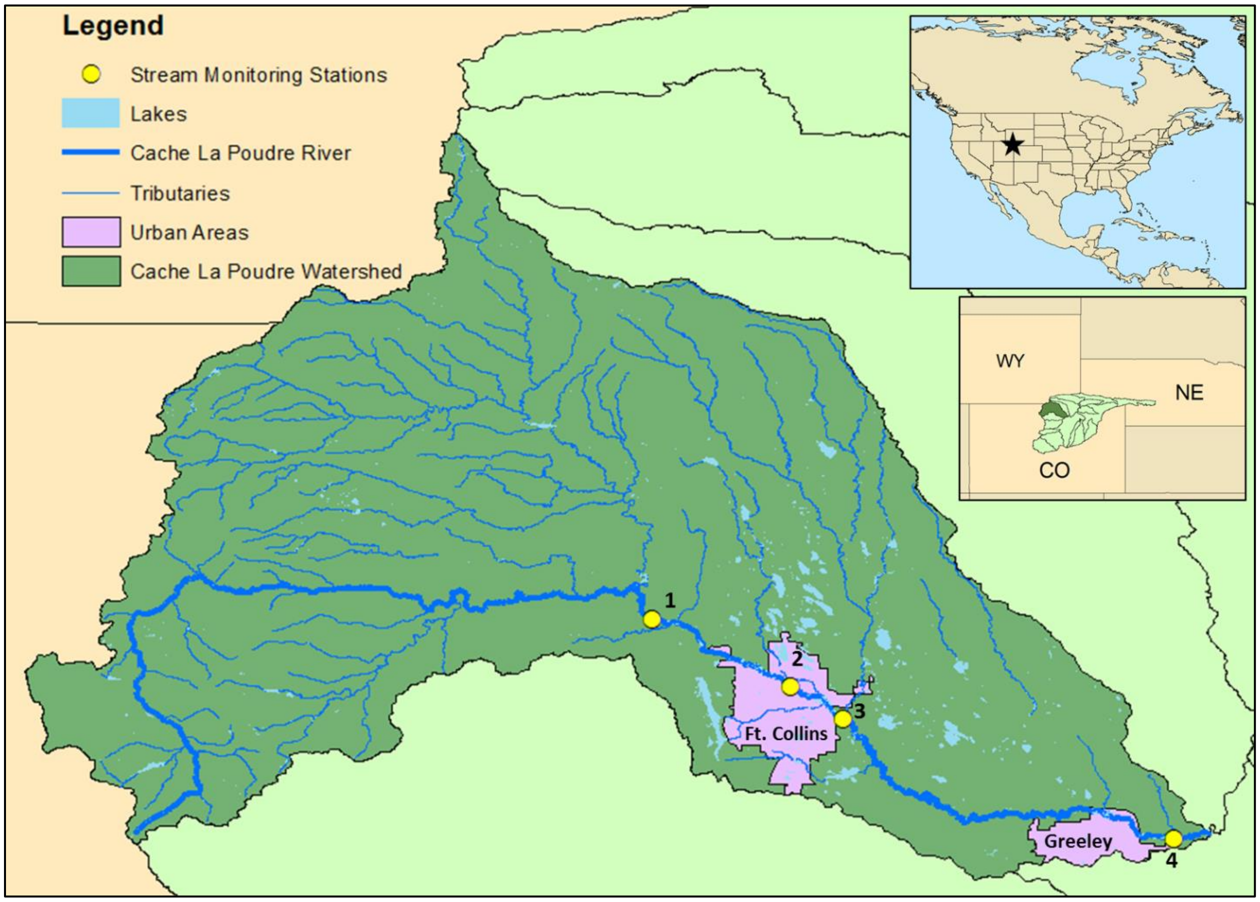


Figure 39: Cache La Poudre River Watershed Location, South Platte River Basin, CO

The Cache La Poudre River (Poudre) is an 1887 square mile watershed in northern Colorado. The Poudre's headwaters original in Rocky Mountain National Park and flows out of the mountains through its canyon before entering the cities of Fort Collins, Windsor, and Greeley, respectively. Upstream of the canyon mouth, Point 1 Figure 39, the basin is relatively undeveloped representing a more natural background state. The lower portion of the watershed is a mix of urban and agricultural land uses before its confluence with the South Platte River downstream of Greeley.

Table 8: CFA's Cache La Poudre River Downstream Analysis Locations

Map Point	Location	USGS Station ID	Flow Data Dates	Nitrogen Data Dates	Comments
1	Mouth of Poudre Canyon	06752000	1/1/1900 – 9/30/2007	7/29/1992 – 8/10/1995	Minimal upstream development, representative of natural background
2	Lincoln Street Fort Collins, CO	06752260	4/8/1975 – 3/23/2014	10/25/1979 – 4/15/1994	Urban drainage and some waste water treatment plant impacts
3	Above Confluence with Boxelder Creek	06752280	10/1/1979 – 3/23/2014	10/24/1979 – 4/15/1994	Contains impacts and treatment plants from the City of Fort Collins
4	Downstream of Greeley, CO	06752500	4/1/1903 – 9/30/1998	7/30/1992 – 1/12/1995	Contains impacts from Fort Collins, Windsor, agriculture, and Greeley

3.3: RESULTS AND DISCUSSION

3.3.1: DOWNSTREAM APPLICATION

The map-based interface for the CFA tool facilitates a "downstream application" of flow analyses using the tool's standardized approach to better understand and compare flow and nutrient dynamics along the length of a river. This was applied to a stretch of the Poudre River for the four points shown in Figure 39, described in Table 8. A load duration curve (LDC) analysis was applied to each of the four locations analyzing available flow data and total nitrogen tests (USGS water quality code 006000) for the stations from USGS' NWIS. For the analysis a target nitrogen concentration of 2 mg/L was used in conjunction with the flow data to determine a total allowable nitrogen load based on flow recurrence interval.

As show in Figure 40, at Point 1, all observed nitrogen concentrations are below the target standard and roughly parallel to the total LDC. The water quality points are also highlighted to reflect which part of the season they occur in, the points highlighted in black were witnessed from April to October. This season of stream flows was selected to highlight the impacts of the annual snow melt hydrograph from the winter seasonal low flows typical for rivers in this area. In addition to the water quality points, a boxplot of the observed nitrogen concentrations is provided for each of the five flow intervals of the duration curve. Additionally, each of the annual LDCs, plotted in grey, are clustered near the red LDC for the complete period of record which indicates minimal variation in stream flows over time, which supports the use of this station as a natural background indicator.

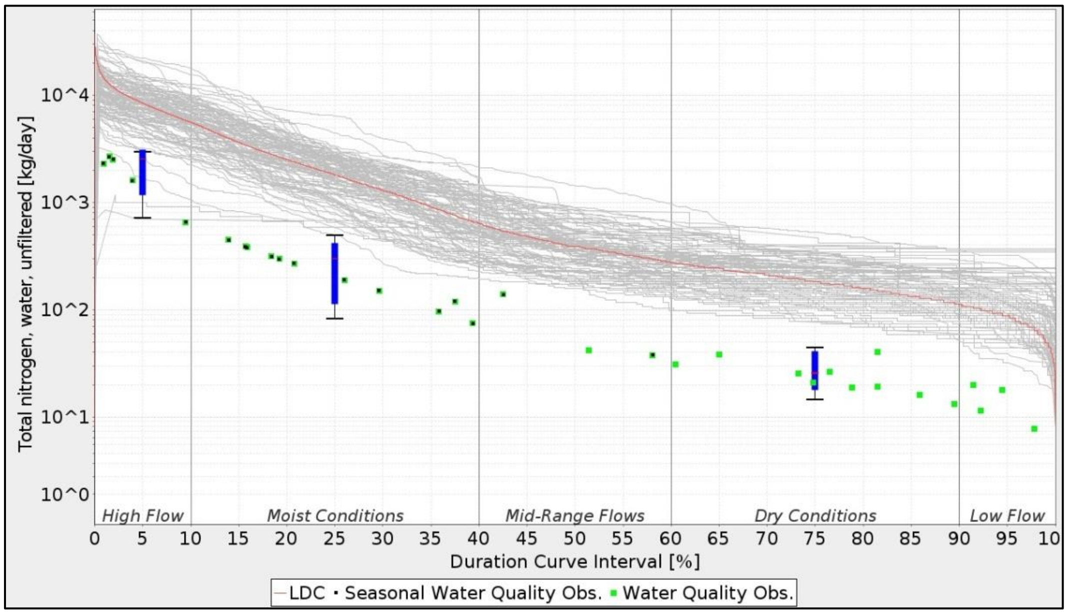


Figure 40: CFA's Load Duration Curve for Point 1, Cache La Poudre River at Mouth of Canyon

At Point 2, in Fort Collins, it is quickly evident that the natural flow regime has been disturbed from urban development due to the wider spread in the annual LDCs in Figure 41. Additionally, nitrogen concentrations are elevated closer to the target concentration than for Point 1. This is due to some agricultural and urban storm water influences which drain to this portion of the river. The shape of the LDC has also changed to reflect the impacts of urbanization on a watershed which is a tendency towards higher peak flows more frequently due to added imperviousness. The smaller magnitude of "Low Flows" portion of the LDC relative to Point 1 is due to water diversions out of the river between the canyon mouth and Fort Collins.

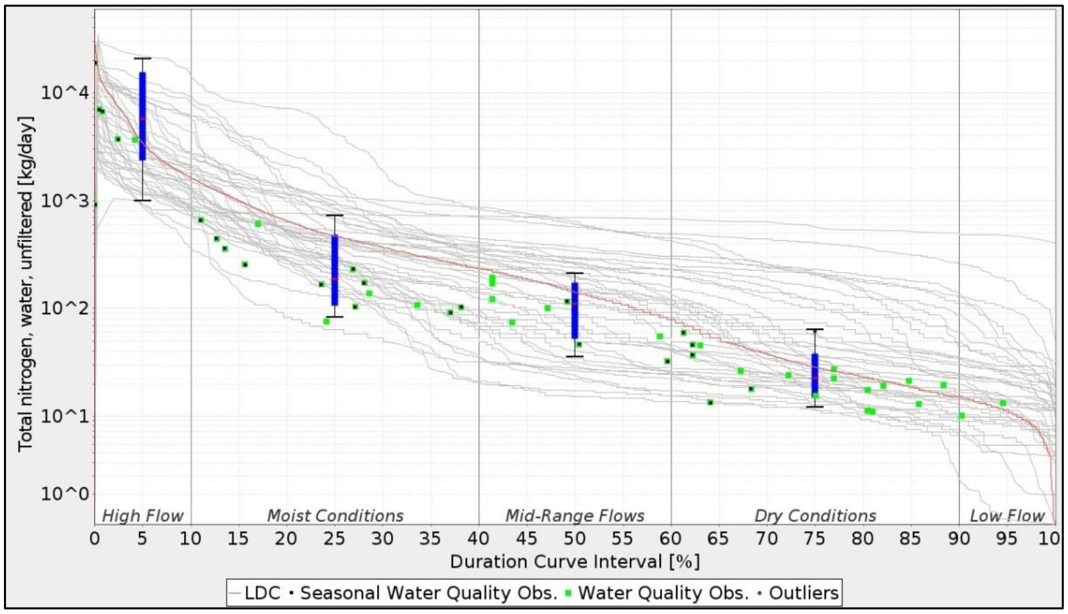


Figure 41: CFA's Load Duration Curve for Point 2, Cache La Poudre River at Fort Collins, CO

Point 3 further illustrates the urban impacts of the City of Fort Collins on the flow regime of the Poudre River. Again, the flood peaks are further heightened and their frequency reduced. The Dry Conditions and Low Flows section remain at a lower magnitude than that of upstream. Additionally, the added complexity and variability of urban drainage and some agricultural return flow impacts is reflected in the wider spread of the annual LDCs. A further illustration of the downstream dynamics of the river as affected by the city, are the elevated nitrogen levels which now regularly exceed the target concentration of 2 mg/L.

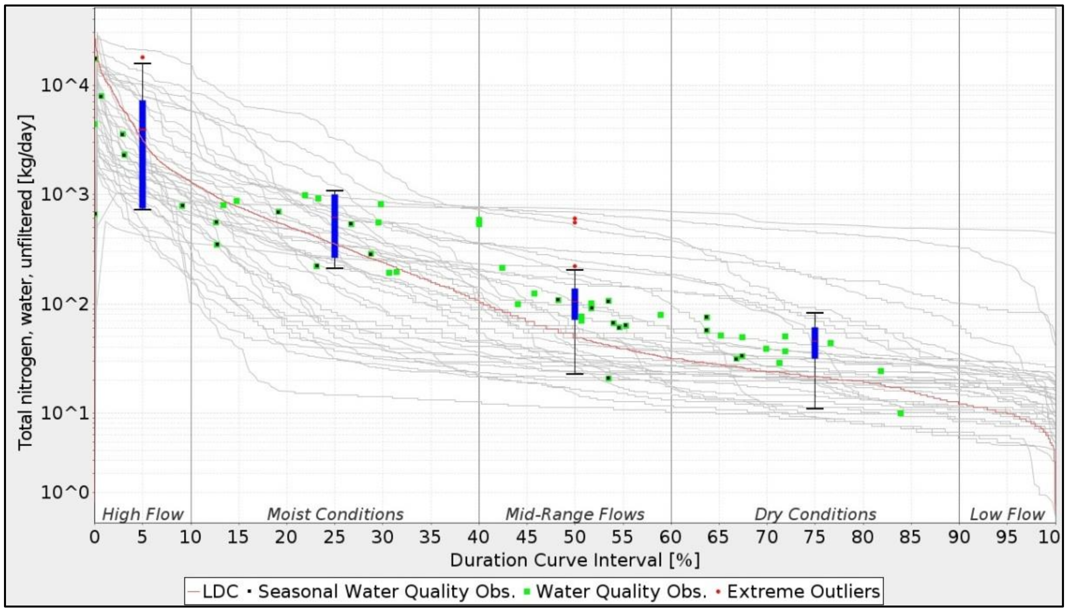


Figure 42: CFA's Load Duration Curve for Point 3, Cache La Poudre River above Boxelder Creek

Point 4, Figure 43, illustrates a very different flow regime of the Poudre River than that before Fort Collins. After Fort Collins and before Greeley, agricultural groundwater return flows have contributed to the river elevating the Dry Conditions and Low Flow sections of the LDC. However, due to the urban drainage of Fort Collins, Windsor, and Greeley, the High Flows remain high. As with the previous points, the overall nitrogen concentration have again increased dramatically above the target concentration illustrating downstream impacts of urban and agricultural areas on this river. A downstream analysis, like this one, can shed light on different influences to a river system and help sort out where things start to change. It can also assist regulating agencies in assessing the viability of future regulation standards against currently available data.

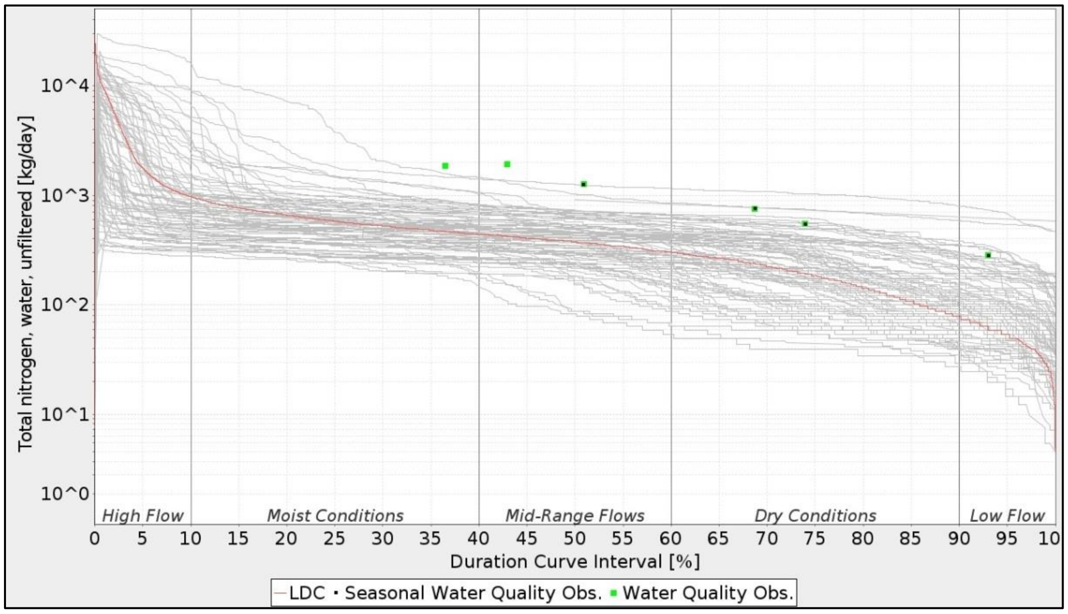


Figure 43: CFA’s Load Duration Curve for Point 4, Cache La Poudre River near Greeley, CO

3.3.2: SCALABILITY

In order to test the expected scalability of the CFA tool, a series of test cases were generated for each of the analyses. These test cases were then sent for execution to a cloud environment using 2-core, extra-large, Amazon-cloud, virtual machines. Based on an understanding of the behind the scenes of CFA’s analyses, a preliminary scalability test was performed using estimated request rates that were expected to stress the scalability of the system. These rates were estimated based primarily on the complexity of the analysis, simple analyses were tested at higher request rates, complex analyses were tested at lower request rates. Table 9 below summarizes the initial scalability request rates for each of the various models in the CFA tool; these rates are approximate in practice due to overhead computational costs of setting up the testing infrastructure and other similar limitations.

Table 9: CFA's Preliminary Scalability Request Rates

CFA Analysis	Preliminary Request Rate
Time Series/Statistics	10 req/sec
Flood (Bulletin 17B)	25 req/sec
Drought (Salas et al., 2005)	2 req/sec
Flow Duration Curve	10 req/sec
Load Duration Curve	2 req/sec
Base-flow Separation (BFLOW)	2 req/sec
Load Estimator (LOADEST)	2 req/sec

The results of this preliminary scalability testing are shown below in Figure 44. The high request rates for time-series/statistics and the flow duration curve illustrate the expected decrease in analysis execution time with increased available infrastructure (VMs). However, the similarly simple load duration curve and drought analyses show almost no improvement with increased number of VMs due to their lower request rates, meaning that the initial infrastructure of the first 4 VMs are likely sufficient to complete the analyses at the given request rates. The irregularity in the flood analysis results was unexpected and examined further. It was discovered that the high request rate of the flood analysis testing coupled with how quickly the flood analyses completed, new VMs were unable to launch properly for the next step of the testing cycle resulting in an irregular execution time versus available infrastructure curve.

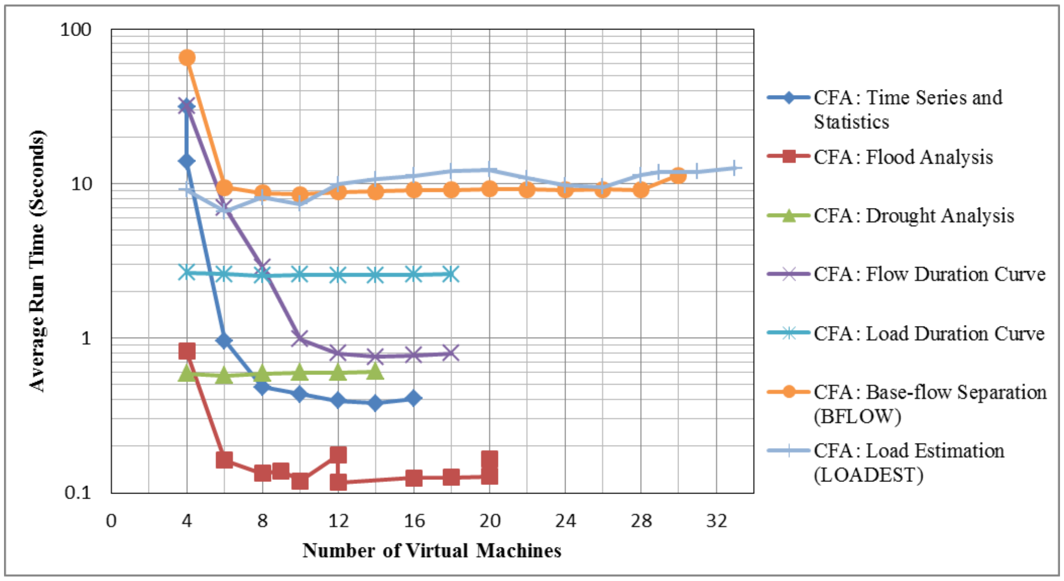


Figure 44: CFA's Preliminary Scalability Testing Results

The base-flow separation (BFLOW) and load estimator (LOADEST) models, due to their greater complexity, were tested over a broader range of number of virtual machines in an attempt to view a more complex response to available infrastructure. However, the preliminary testing, Figure 44, revealed that after an initial amount of VMs are supplied to those particular models, minimal or negative additional computational benefit is gained with the addition of more virtual machines. Additionally, after preliminary testing it was found that only 37% of the LOADEST test cases ran to completion. Upon a re-examination of the test cases, the majority of the LOADEST test cases were found to be faulty and were replaced and re-verified before any additional scalability testing to ensure proper results. Then the LOADEST and BFLOW analyses were re-tested for scalability at a variety of new request rates in an attempt to better stress the scalability of these particular models. Only the load estimation and base-flow separation analyses were tested because based on the preliminary testing these were the only analysis in CFA that appeared to have a significant computational burden.

Shown below in Figure 45, the scalability results for the load estimation illustrate a general trend that with increased cyberinfrastructure, number of VMs, and the average runtime decreases. Additionally, as request rates are increased, there is a trend to increase the average run time. However, for this particular analysis the additional benefit of more than 12 virtual machines does not appear to assist with quicker model execution. Load estimation tests converged to an average value of about 7 seconds after only 12 virtual machines became available.

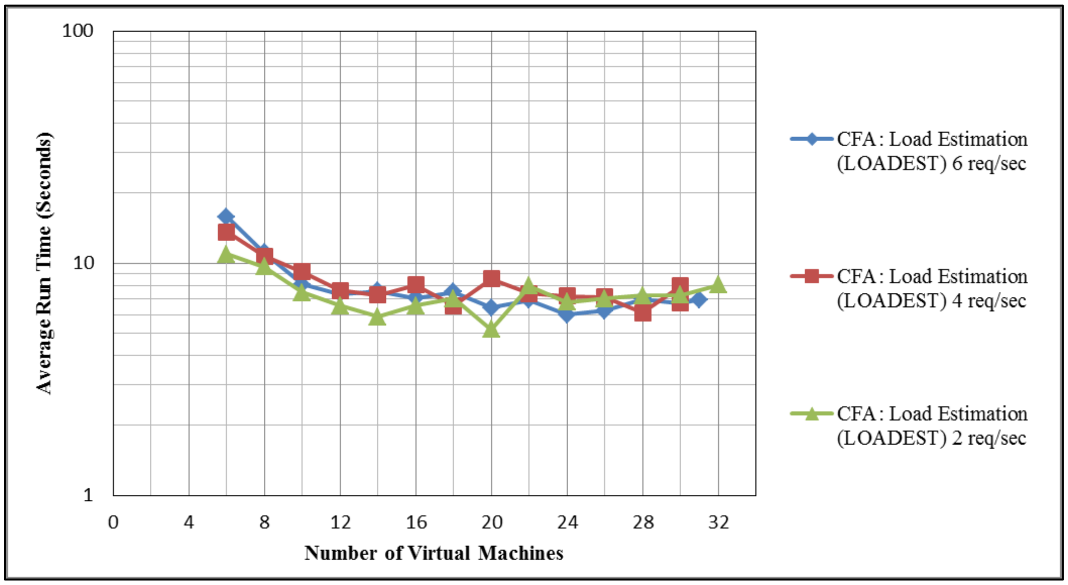


Figure 45: CFA's Additional Load Estimator (LOADEST) Scalability Testing Results

The scalability results for the base-flow separation were more complex. As shown below in Figure 46, the BFLOW analysis appeared to reach a much more stable average execution time for its models, approximately 8 seconds despite having different request rates. The base-flow separation analysis also showed the same increase in execution time with increased request rate for a given set of initial VMs. Both the load estimator and base-flow separation analyses support the claim of scalability with their decreased execution time for both greater available infrastructure (number of VMs) and for lower request rates.

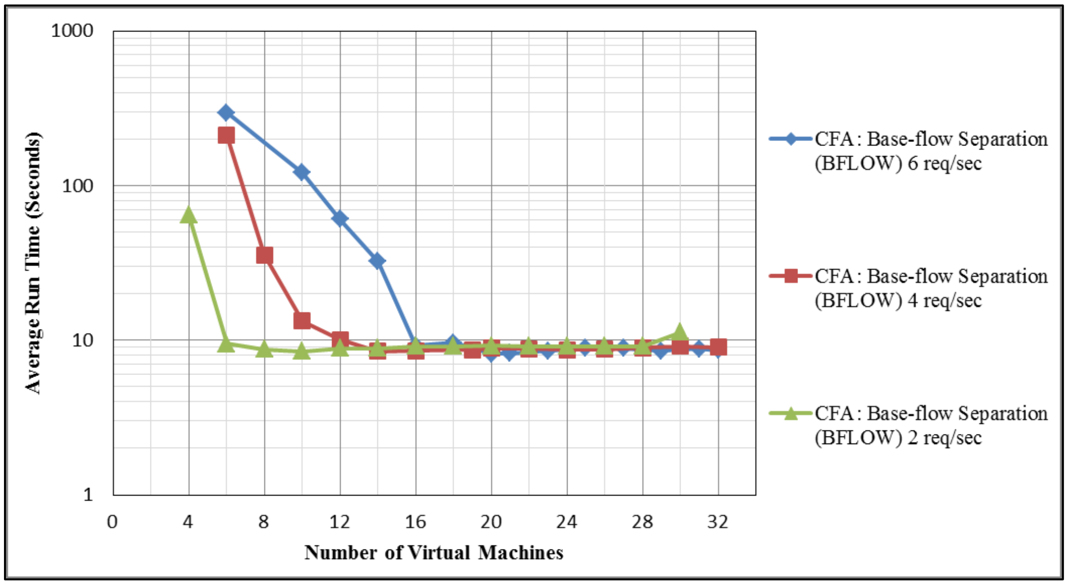


Figure 46: CFA's Additional Base-flow Separation (BFLOW) Scalability Testing Results

3.3.3: LIMITATIONS

One issue that the scalability testing revealed is that some of the analyses in CFA; FDC, LDC, flood, and drought analysis, are very simple. This in turn requires a very large, possibly unrealistic, web-request rate to identify and illustrate scalability benefits with cloud-based VMs in CSIP. Even on a globally available website with thousands of users, a request rate 25 requests per second is probably not realistic; thus the scalability benefits of the simple CFA analyses are likely minimal. The flood analysis testing also illuminated a bottle neck of the current implementation of the infrastructure to launch new VMs for testing due to the quick execution of the analyses. However, due to how quickly the analyses completed it is likely unnecessary to launch new VMs to handle an increase in this particular analyses' demand because the current infrastructure will be available to handle the requests in a very short amount of time anyway.

3.4: CONCLUSIONS

Water will continue to be an important component in cities, agriculture, and industry. To better understand existing stream systems requires more tools and analysis techniques that need to be easily available and make use of recent advancements in computer technology. As tools become more complex, proper management of computational resources become more important. A current solution to proper management of cyberinfrastructure is the use and development of flow analysis software using cloud-based cyberinfrastructure to combine various analyses into single tool and provide a scalable modeling infrastructure. A single unified tool requires data to be preprocessed once rather than once for each analysis and software package being used resulting in more efficient use of time and resources. Additionally, web support for flow analyses will continue to be an important step in use of a tool; otherwise limitations like access to complicated inputs, like regional flood skewness coefficients, will remain beyond the reach of many users. The Comprehensive Flow Analysis (CFA) tool provides this software package unity allowing multiple flow analyses on a single formatted dataset. Furthermore, the scalability of the various CFA flow analyses under different cyberinfrastructure hardware configurations and request rates demonstrates that this is a scalable tool. The scalable infrastructure combined with the database access built into CFA creates a more streamlined analysis process from data collection to analysis to results.

CHAPTER 4: CONCLUSIONS

The complex issues of water demand and environmental change in watersheds have driven the desire to simulate and model watersheds and the impacts of various management and climate change scenarios. The pseudo-distributed watershed model SWAT and the finite-difference groundwater model MODFLOW are at the forefront of popular available watershed models. Inabilities of these models to simulate complex groundwater response as well as erosion, plant growth, nutrient cycling, and agricultural management has led to multiple attempts to link the models. However, these models are unavailable and have drawbacks like required identical spatial discretization and monthly coupling. As a result the aforementioned work was undertaken to create a generalized spatial linkage between SWAT and MODFLOW which retains the respective strengths of the two models coupled on a daily time-step. This linkage facilitates the use of existing spatially dis-similar SWAT and MODFLOW models while increasing the overall quality of simulation for a more reliable result than previously possible.

This daily coupling between SWAT and MODFLOW has resulted in realistic and accurate stream flow results for the North Fork of the Sprague River in Oregon. Additionally, groundwater representation within SWAT was increased by the inclusion of MODFLOW and accurate groundwater table elevations were simulated as a result. However, at the high elevation edge of the watershed a reduced volume of recharge to the aquifer caused lower water table elevations relative to the original MODFLOW model for the Upper Klamath Basin. The new SWAT-MODFLOW coupling resulted in a more accurate representation of both the frequency and magnitude of streams flows than the original calibrated SWAT model only. This model coupling also provided these results without additional calibration. It is likely calibration of the coupled SWAT-MODFLOW model will improve all the results herein discussed.

Additionally, as better ways to access and implement flow analyses become available it is important to leverage these advances. Water will continue to be an important component in cities, agriculture, and industry. To better understand existing stream systems requires more tools and analysis techniques that need to be easily available and make use of recent advancements in computer technology. As tools become more complex, proper management of computational resources become more important. A current solution to proper management of cyberinfrastructure is the use and development of flow analysis software using cloud-based cyberinfrastructure to combine various analyses into single tool and provide a scalable modeling infrastructure. A single unified tool requires data to be preprocessed once rather than once for each analysis and software package being used resulting in more efficient use of time and resources. Additionally, web support for flow analyses will continue to be an important step in use of a tool; otherwise limitations like access to complicated inputs, like regional flood skewness coefficients, will remain beyond the reach of many users. The Comprehensive Flow Analysis (CFA) tool provides this software package unity allowing multiple flow analyses on a single formatted dataset. Furthermore, the scalability of the various CFA flow analyses under different cyberinfrastructure hardware configurations and request rates demonstrates that this is a scalable tool. The scalable infrastructure combined with the database access built into CFA creates a more streamlined analysis process from data collection to analysis to results.

REFERENCES

Ahmadi, M., R. Records, and M. Arabi. 2013. Impacts of climate change on diffuse pollutant fluxes at the watershed scale. Hydrologic Process. Published online. doi: 10.1002/hyp.9723.

Arnold, J. G., and N. Fohrer. 2005. SWAT2000: Current capabilities and research opportunities in applied watershed modeling. Hydrological Processes. 19(3): 563-572.

Arnold, J.G. and P.M. Allen. 1999a. Automated methods for estimating baseflow and ground water recharge from streamflow records. Journal of the American Water Resources Association 35(2): 411-424.

Arnold, J. G., R. Srinivasan, R. S. Muttiah, P. M. Allen, and C. Walker. 1999b. Continental-scale simulation of the hydrologic balance. Journal of American Water Resources Association. 35(5): 1037-1052.

Arnold, J.G., Srinivasan, R., Muttiah, R.S., Williams, J.R. 1998. Large area hydrologic modeling and assessment Part I Model development. Journal of American Water Resources Association. (JAWRA), 34(1): 73-89.

Arnold, J. G., P. M. Allen, R. Muttiah, & G. Bernhardt. 1995a. Automated base flow separation and recession analysis techniques. Ground Water, 33(6): 1010-1010.

Arnold, J. G., J. R. Williams, and D. R. Maidment. 1995b. Continuous-time water and sediment-routing model for large basins. Journal of Hydrologic Engineering ASCE 121(2): 171-183.

Arnold, J. G., and J. R. Williams. 1987. Validation of SWRRB: Simulator for water resources in rural basins. Journal of Water Resources Planning and Management ASCE 113(2): 243-256.

Atkins, Jr., J.T., J.B. Wiley, and K.S. Paybins. 2009. Generalized Skew Coefficients of Annual Peak Flows for Rural, Unregulated Streams in West Virginia. U.S. Geological Survey: Open-File Report: 2008-1304.

Baffaut, C., and V.W. Benson. 2009. Modeling Flow and Pollutant Transport in a Karst Watershed with SWAT. Trans. ASABE 52(2): 469-479.

Bailey, R.T., Gates, T.K., and M. Ahmadi. 2014. Simulating reactive transport of selenium coupled with nitrogen in a regional-scale irrigated groundwater system. Journal of Hydrology, accepted.

Bailey, R.T., T.K. Gates, and A.D. Halvorson. 2013a. Simulating variably-saturated reactive transport of selenium and nitrogen in agricultural groundwater systems. Journal of Contaminant Hydrology, 149, 27-45.

Bailey, R.T., E.D. Morway, R. Niswonger, and T.K. Gates. 2013b. Modeling variably saturated multispecies reactive groundwater solute transport with MODFLOW-UZF and RT3D. Groundwater, 51(5), 752-761.

Bailey, R.T., J.H. William, and T.K. Gates. 2012. The Influence of Nitrate on Selenium in Irrigated Agricultural Groundwater Systems.

Beven, Keith, R. Lamb, P. Quinn, R. Romanowicz, and J. Freer (V. P. Singh). "Topmodel." Computer models of watershed hydrology. 1995: 627-668.

Booth, N.L., E.J. Everman, I. Kuo, L. Sprague, and L. Murphy. 2011. A Web-based Decision Support System for Assessing Regional Water-Quality Conditions and Management Actions. Journal of the American Water Resources Association 47(5): 1136-1150.

Borah, D. K., G. Yagow, A. Saleh, P. L. Barnes, W. Rosenthal, E. C. Krug, and L. M. Hauck. 2006. Sediment and nutrient modeling for TMDL development and implementation. Trans. ASABE 49(4): 967-986.

Brown, L. C., and T. O. Barnwell, Jr. 1987. The enhanced water quality models QUAL2E and QUAL2E-UNCAS: Documentation and user manual. EPA document EPA/600/3-87/007. Athens, Ga.: USEPA.

Burger C.M., S. Kollet, J. Schumacher, and D. Bosel. 2012. Introduction of a web service for cloud computing with the integrated hydrologic simulation platform ParFlow. Computers and Geosciences 48(2012): 334-336.

Camporese, M., C. Paniconi, M. Putti, and S. Orlandini. 2010. Surface-subsurface flow modeling with path-based runoff routing, boundary condition-based coupling, and assimilation of multisource observation data. Water Resources Research. 46:W022512, doi:10.1029/2008WR007536.

CHESS. 2001. Climate, hydrochemistry, and economics of surface-water systems. Available at: www.nwl.ac.uk/ih/www/research/images/chessreport.pdf. Accessed 6 February 2014.

Christenson, S., N.I. Osborn, C.R. Neel, J.R. Faith, C.D. Blome, J. Puckette, and M.P. Pantea. 2011. Hydrogeology and Simulation of the Groundwater Flow in the Arbuckle-Simpson Aquifer, South-Central Oklahoma. U.S. Geological Survey Scientific Investigations Report 2011-5029.

Chu, T. W., and A. Shirmohammadi. 2004. Evaluation of the SWAT model's hydrology component in the Piedmont physiographic region of Maryland. Trans. ASAE 47(4): 1057-1073.

Chung, I., N. Kim, J. Lee, and M. Sophocleous. 2010. Assessing distributed groundwater recharge rate using integrated surface water-groundwater modelling: application to Mihocheon watershed, South Korea. Hydrogeology Journal 18: 1253-1264.

Cleland, Bruce. August 2007. "An Approach for Using Load Duration Curves in the Development of TMDLs."National TMDL Science and Policy.

Cleland, Bruce. November 2003. "TMDL Development from the 'Bottom Up' - Part III: Duration Curves and Wet-Weather Assessments." National TMDL Science and Policy.

Cleland, Bruce. August 2002. “TMDL Development from the ‘Bottom Up’ Part II: Using Duration Curves to Connect the Pieces.” National TMDL Science and Policy 2002.

Clement, T.P., Y. Sun, B.S. Hooker, and J.N. Petersen. 1998. Modeling Multispecies Reactive Transport in Ground Water. Groundwater Monitoring & Remediation 18(2): 79-92.

Conan, C., F. Bouraoui, N. Turpin, G. de Marsily, and G. Bidoglio. 2003. Modeling flow and nitrate fate at catchment scale in Brittany (France). Journal of Environmental Quality 32(6):2026–2032.

Cooper, R.M., 2005. Estimation of Peak Discharges for Rural, Unregulated Streams in Western Oregon. U.S. Geological Survey: Scientific Investigations Report 2005-5116.

Council, G. W. 1999. A Lake Package for MODFLOW (LAK2): Documentation and User's Manual, Version 2.2. HIS Geotrans, Sterling, Virginia, USA.

Ferguson, I.M., and R.M. Maxwell. 2010. Role of groundwater in watershed response and land surface feedbacks under climate change. Water Resources Research. 46:W00F02, doi:10.1029/2009WR008616.

Flynn, K.M., W.H. Kirby, and P.R. Hummel. 2006. User's Manual for Program PeakFQ, Annual Flood-Frequency Analysis Using Bulletin 17B Guidelines. U.S. Geological Survey: Techniques and Methods 4-B4.

Galbiati L, F. Bouraoui, F.J. Elorza, and G. Bidoglio. 2006. Modeling diffuse pollution loading into a Mediterranean lagoon: development and application of an integrated surface-subsurface model tool. Ecological Modeling. 193(1–2):4–18.

Gannett, M.W., B.J. Wagner, and K.E. Lite Jr. 2012. Groundwater simulation and management models for the Upper Klamath Basin, Oregon and California. U.S. Geological Survey Scientific Investigations Report 2012-5062.

Gannett, M.W., K.E. Lite, Jr., J.L. La Marche, B.J. Fisher, and D.J. Polette. 2010. Groundwater hydrology of the Upper Klamath Basin, Oregon and California. U.S. Geological Survey Scientific Investigations Report 2007-5050, Version 1.1.

Gassman, P.W., M.R. Reyes, C.H. Green, J.G. Arnold. 2007. The Soil and Water Assessment Tool: Historical Development, Applications, and Future Research Directions. Trans. ASABE: 50(4): 1211-1250.

Gobindaraju, R.S., B. Engel, D. Ebert, B. Fossum, M. Huber, C. Jafvert, S. Kumar, V. Merwade, D. Niyogi, L. Loliver, S. Prabhakar, G. Rochon, C. Song, and L. Zhao. 2009. Vision of Cyberinfrastructure for End-to-End Environmental Exploration (C4E4). Journal of Hydrologic Engineering American Society of Civil Engineers, 12, 1:56-64.

Harbaugh A.W. 2005. MODFLOW-2005, The U.S. Geological Survey Modular Ground-Water Model-The Ground-Water Flow Process. USGS Techniques and Methods 6-A16.

Harbaugh, A.W., E.R. Banta, M.C. Hill, and M.G. McDonald. 2000. MODFLOW-2000, The U.S. Geological Survey modular ground-water model – User guide to modularization concepts and the ground-water flow process. USGS Open-File Report: 2000-92.

Hargreaves, G. L., G. H. Hargreaves, and J. P. Riley. 1985. Agricultural benefits for Senegal River basin. J. Irrig. Drain. Eng. 108(3): 225-230.

Hay, L.E., S.L. Markstrom, and C. Ward-Garrison. 2011. Watershed-scale response to climate change through the twenty-first century for selected basins across the United States. Earth Impact. 15(17): 1-37.

Interagency Advisory Committee on Water Data (IACWD). 1982. "Guidelines for determining flood flow frequency." Bulletin No. 17B (revised and corrected), Hydrology Subcommittee, Washington, D.C.

Izaurralde, R. C., J. R. Williams, W. B. McGill, N. J. Rosenberg, and M. C. Quiroga Jakas. 2006. Simulating soil C dynamics with EPIC: Model description and testing against long-term data. Ecological Modeling 192(3-4): 362-384.

Jeppsen, E., B. Kronvang, M. Meerhoff, M. Sondergaard, K.M. Hansen, H.E. Andersen, T.L. Lauridsen, L. Liboriussen, M. Beklioglu, A. Ozen, and J.E. Olesen. 2007. Climate change effects on runoff, catchment phosphorus loading and lake ecological state, and potential adaptations. Journal of Environmental Quality. 38(5): 1930-41.

Kim NW, I.M. Chung, Y.S. Won, J.G. Arnold. 2008. Development and application of the integrated SWAT-MODFLOW model. Journal of Hydrology 356:1–16.

Knisel, W. G. 1980. CREAMS, a field-scale model for chemicals, runoff, and erosion from agricultural management systems. USDA Conservation Research Report No. 26. Washington, D.C.: USDA.

Kollet, S.J., and R.M. Maxwell. 2006. Integrated surface-groundwater flow modeling: A free-surface overland flow boundary condition in a parallel groundwater flow model. Advances in Water Resources 29: 945-985.

Leonard, R. A., W. G. Knisel, and D. A. Still. 1987. GLEAMS: Groundwater loading effects of agricultural management systems. Trans. ASAE 30(5): 1403-1418.

Lettenmaier, D.P., A.W. Wood, R.N. Palmer, E.F. Wood, and E.Z. Stakhiv. 1999. Water resources impacts of global warming: A US regional perspective. Climate Change 43(3): 537-579.

Lim K.J., B.A. Engel, Z. Tang, J. Choi, K. Kim, S. Muthukrishnan, and D. Tripathy. 2005. Automated Web GIS Based Hydrograph Analysis Tool, WHAT. Journal of the American Water Resources Association 41(6): 1407-1416.

Lloyd, W., O. David, J. Lyon, K.W. Rojas, J.C. Ascough II, T.R. Green, and J.R. Carlson. 2012. The Cloud Services Innovation Platform Enabling Service-Based Environmental Modelling Using Infrastructure-as-a-Service Cloud Computing. International Environmental Modelling and Software Society 2012 Conference Proceedings.

Luo, Y., and M. Sophocleous. 2011. Two-way coupling of unsaturated-saturated flow by integrating the SWAT and MODFLOW models with application in an irrigation district in arid region of West China. Journal of Arid Land. 3(3): 164-173. doi: 10.3724/SP.J.1227.2011.00164.

Markstrom, S.L., R.G. Niswonger, R.S. Regan, D.E. Prudic, and P.M. Barlow. 2008. GSFLOW-Coupled Ground-Water and Surface-Water Flow Model Based on the Integration of the Precipitation-Runoff Modeling System (PRMS) and the Modular Ground-Water Flow Model (MODFLOW-2005). U.S. Geological Survey Techniques and Methods 6-D1: 240p.

Mashburn, S.L., D.W. Ryter, C.R. Neel, S.J. Smith, and J.S. Magers. 2013. Hydrogeology and Simulation of Groundwater Flow in the Central Oklahoma (Garber-Wellington) Aquifer, Oklahoma, 10987 to 2009, and Simulation of Available Water in Storage, 2010-2059. U.S. Geological Survey Scientific Investigations Report 2013-5219.

McDonald, M.G., and A.W. Harbaugh.1988. A modular three-dimensional finite-differences ground-water flow model. U.S. Geological Survey. Techniques of Water-Resources investigations Book 6, Ch. Al., p. 596.

Menking, K. M., R. Y, Anderson, N. G. Shafike, K. H. Syed, and B. D. Allen. 2004. Wetter or colder during the last glacial maximum? Revisiting the pluvial lake question in southwestern North American Quaternary Research. 62(3): 280-288.

Menking K.M., K.H. Syed, R.Y. Anderson, N.G. Shafike, and J.G. Arnold. 2003. Model estimates of runoff in the closed, semiarid Estancia basin, central New Mexico, USA. Hydrological Sciences Journal 48(6):953–970.

Monteith, J. L. 1965. Evaporation and the environment. In The State and Movement of Water in Living Organisms, Proc. 19th Symp. Swansea, U.K.: Society of Experimental Biology, Cambridge University Press.

Morway, E.D., T.K. Gates, and R.G. Niswonger. 2013. Appraising Options to Reduce Shallow Groundwater Tables and Enhance Flow Conditions Over Regional Scales in an Irrigated Alluvial Aquifer System. Journal of Hydrology 495(2013): 216-237.

Neitsch, S. L., J. G. Arnold, J. R. Kiniry, and J. R. Williams. 2011. Soil and Water Assessment Tool Theoretical Documentation, Version 2009. Temple, Tex.: Texas Water Resources Institute Technical Report No. 406. Available at: http://twri.tamu.edu/reports/2011/tr406.pdf Accessed 7 February 2014.

Neitsch, S. L., J. G. Arnold, J. R. Kiniry, and J. R. Williams. 2005. Soil and Water Assessment Tool Theoretical Documentation, Version 2005. Temple, Tex.: USDA-ARS Grassland, Soil and Water Research Laboratory. Available at: http://swat.tamu.edu/media/1292/SWAT2005theory.pdf. Accessed 7 February 2014.

Niswonger, R.G., S. Panday, and M. Ibaraki. 2011. MODFLOW-NWT, A Newton formulation for MODFLOW-2005: USGS Survey Techniques and Methods 6–A37.

Niswonger, R.G., D.E. Prudic, and R.S. Regan. 2006. Documentation of the unsaturated-zone flow (UZF1) package for modeling unsaturated flow between the land surface and the water table with MODFLOW-2005, USGS Techniques and Methods 6-A19.

Olson, S.A., 2009. Estimation of Flood Discharges at Selected Recurrence Intervals for Streams in New Hampshire. U.S. Geological Survey: Scientific Investigations Report 2008-5206.

Parrett, C., D.R. Johnson. 2004. Methods for Estimating Flood Frequency in Montana Based on Data through Water Year 1998. U.S. Geological Survey: Water-Resources Investigations Report 03-4308.

Paschke, S.S. 2011. Groundwater Availability of the Denver Basin Aquifer System, Colorado. U.S. Geological Survey Professional Paper 1770.

Perkins, S.P., and M. Sophocleous. 1999. Development of a comprehensive watershed model applied to study stream yield under drought conditions. Ground Water 37(3):418–426.

Peterson, J. R., and J. M. Hamlet. 1998. Hydrologic calibration of the SWAT model in a watershed containing fragipan soils. Journal of American Water Resources Association 34(3): 531-544.

Pomeroy, C.A., and A.J. Timpson. 2010. Methods for Estimating Magnitude and Frequency of Peak Flows for Small Watersheds in Utah. U.S. Geological Survey: Report No UT-10.11.

Priestly, C. H. B., and R. J. Taylor. 1972. On the assessment of surface heat flux and evaporation using large-scale parameters. Monthly Weather Rev. 100(2): 81-92.

Prommer, H., D.A. Barry, and C. Zheng. 2003. MODFLOW/MT3DMS-Based Reactive Multicomponent Transport Modeling. Groundwater 42(2): 247-257.

Prudic, D.E., L.F. Konikow, and E.R. Banta. 2004. A New Streamflow Routing (SFR1) Package to Simulate Stream-Aquifer Interaction with MODFLOW-2000. U.S. Geological Survey Open-File Report 2004-1042.

Rathjens, H., and N. Oppelt. 2012. SWATgrid: An interface for setting up SWAT in a grid-based discretization scheme. Computers & Geosciences. 45: 161-167.

Records, R. 2013. Water Quality Benefits of Wetlands under Historic and Potential Future Climate in the Sprague River Watershed, Oregon. MS Thesis, Colorado State University, Fort Collins, CO.

Rumman, M.A., and D.F. Payne. 2003. Model Framework and Preliminary Results of the Regional MODFLOW Ground-Water Flow Model of Costal Georgia, South Carolina, and Florida. 2003. Proceedings of the 2003 Georgia Water Resources Conference held April 23-24, 2003, at the University of Georgia.

Runkel, R.L., C.G. Crawford, and T.A. Cohn. 2004. Chapter A5: Load Estimator (LOADEST): A FORTRAN Program for Estimating Constituent Loads in Streams and Rivers. U.S. Geological Survey Techniques and Methods Book 4.

Seaber, P. R., F. P. Kapinos, and G. L. Knapp. 1987. Hydrologic units maps. USGS Water-Supply Paper No. 2294. Reston, Va.: U.S. Geological Survey.

Sloto, R.A., and M.Y. Crouse. 1996. HYSEP: A Computer Program for Streamflow Hydrograph Separation and Analysis: U.S. Geological Survey Water-Resources Investigations Report 96-4040, 46 p.

Solheim, A.L., K. Austnes, T.E. Eriksen, I. Seifert, and S. Holen. 2010. Climate change impacts on water quality and biodiversity: Background report for EEAA European Environment State and Outlook Report 2010 ETC Water Technical Report 1/2010. Prague, Czech Republic.

Soong, D.T., A.L. Ishii, J.B. Sharpe, and C.F. Avery. 2004. Estimating Flood-Peak Discharge Magnitudes and Frequencies for Rural Streams in Illinois. U.S. Geological Survey: Scientific Investigations Report 2004-5103.

Sophocleous, M., and S.P. Perkins. 2000. Methodology and application of combined watershed and ground-water models in Kansas. Journal of Hydrology 236(3–4):185–201.

Spruill, C. A., S. R. Workman, and J. L. Taraba. 2000. Simulation of daily and monthly stream discharge from small watersheds using the SWAT model. Trans. ASAE 43(6): 1431-1439.

Srivastava, P., J. N. McNair, and T. E. Johnson. 2006. Comparison of process-based and artificial neural network approaches for streamflow modeling in an agricultural watershed. Journal of American Water Resources Association 42(2): 545-563.

USDA-NRCS. 2004. Part 630: Hydrology. Chapter 10: Estimation of direct runoff from storm rainfall: Hydraulics and hydrology: Technical references. In NRCS National Engineering Handbook. Washington, D.C.: USDA National Resources Conservation Service. Available at:

http://directives.sc.egov.usda.gov/OpenNonWebContent.aspx?content=17752.wba. Accessed 4 April 2014.

Vallabhaneni, S., CC. Chan, and S.J. Campbell. 2012. SSOAP Toolbox Enhancements and Case Study. U.S. EPA 600/R-12/690.

Water Resources Council, Hydrology Committee (WRC-HC). 1967. "A Uniform Technique for Determining Flood Flow Frequencies." Bulletin No. 15, Washington, D.C.

Williams, J. R., and H. D. Berndt. 1977. Sediment yield prediction based on watershed hydrology. Trans. ASAE 20(6): 1100-4.

Wood, A.W., L.R. Leung, V. Sridhar, and D.P. Lettenmaier. 2004. Hydrologic implications of dynamical and statistical approaches to downscaling climate model outputs. Climate Change 62(1-3): 189-216.

Zheng, C., and P.P. Wang. 1999. MT3DMS: A Modular Three-Dimensional Multispecies Transport Model for Simulation of Advection, Dispersion, and Chemical Reactions of Contaminants in Groundwater Systems; Documentation and User’s Guide. U.S. Army Corps of Engineers Contract Report SERDP-99-1.

Zheng, C. 1990. MT3D, A modular three-dimensional transport model for simulation of advection, dispersion and chemical reactions of contaminants in groundwater systems. Report to the U.S. Environmental Protection Agency, Robert S. Kerr Environmental Research Laboratory, Ada, OK.

APPENDIX I: SWAT-MODFLOW LINKAGE INPUT FILE INSTRUCTIONS

ARCGIS SHAPEFILE MANIPULATION

1. Modify the MODFLOW grid shapefile (FISHNET can be used to create a gridded shapefile in ArcGIS, http://arcscripts.esri.com/details.asp?dbid=12807)
    a. Add new field (grid_id) integer
        i. See "HOW TO SORT AND NUMBER GRID CELLS FOR EXPECTED GRID ID NUMBER SEQUENCE" on how to populate this field
    b. Add new field (Col_Row) string
        i. Populate the field by the column index of the grid cell then a space then a dash then a space then the row "index of the grid cell
        ii. Example: The grid cell in column 71 and row 33 would have Col_Row = "71 – 33"
2. Copy "reach" shapefile (name it reach_new) then:
    a. Add new field (riv_id) integer, which is the river id
        i. Populate as "field calculator" as = SUBBASIN
3. Determine the SWAT/MODFLOW river interaction
    a. Intersect reach_new with the MODFLOW grid shapefile (name it rivergrid)

   b. After intersecting add a new field (rgrid_len) float, the length of each river segment within the MODFLOW grid

      i. populate the field by "calculate geometry"

      ii. Note: this needs to be in units of meters

   c. Export this attribute table as a csv file (name it rivergrid.csv)

4. Copy FullHRU and name it “FullHRU_edit” and modify it

   a. add new field (riv_id) integer, the river this HRU drains to

      i. Populate the field by "field calculator" = SUBBASIN

      ii. Or by the river IDs that you have previously created (note: only 1 river per subbasin)

   b. add new field (hru_id) integer

      i. First list the attribute table in increasing order by the column of "HRUgis"

      ii. Populate by numbering this new field 1,2,...,n this should result such that "HRU 1" is the first HRU in subbasin 1

   c. Take FullHRU_edit _sort and apply the GIS operation "Multipart to singlepart" to get a layer (name it FullDHRU) of disaggregated HRUs (DHRU)

5. Modify the FullDHRU shapefile

   a. add new field (DHRU_area) float

        i. Populate the field by "calculate geometry"

    b. add new field (DHRU_id) long integer

        i. populate the field by "field calculator" as = FID + 1

    c. export attribute table (name it FullDHRU.csv)

6. Intersect the MODFLOW grid shapefile (AllCells_sort) with FullDHRU (name it DHRU_grid)

    a. add new field (overlap_a) float, the overlap area between DHRUs and grid cells

        i. Populate the field by "calculate geometry"

    b. add new field (grid_area) float, the area of a grid cell

        i. Populate the field by "field calculator" as = the area of the MODLFOW grid cell, in meters

        ii. Example: A 250 meter by 250 meter grid would have area = 250*250 = 62500 square meters

    c. Export attribute table as a csv file (name it DHRU_grid.csv)

7. Run createSWAT_MODFLOW_files.java

    a. See Appendix II

HOW TO SORT AND NUMBER GRID CELLS FOR EXPECTED GRID ID NUMBER SEQUENCE

1. Open the Geoprocessing > Python window

2. Copy the below code after filling in the correct file path and file name for your database on the env.workspace line (replace “C:/Projects/mydatabase.mdb” with your file path and name of your database)

```
import arcpy
from arcpy import env
env.workspace = "U:/ArcGIS/Default.mdb"
arcpy.Sort_management("AllCells", "AllCells_sort", [["row", "ASCENDING"], ["col",
"ASCENDING"]], "PEANO")
```

3. Then Modify the AllCells_sort shapefile:

    a. Add new field (grid_id) long int

    b. Open the Field Calculator to populate this field as equal to objectID

APPENDIX II: AUTOMATED SWAT-MODFLOW LINKAGE INPUT FILE CREATION JAVA CODE

CREATESWAT_MODFLOW_FILES.JAVA

```
package SWAT_MODFLOW;

import java.io.IOException;
import javax.swing.JOptionPane;

/**
* Last Updated: 28-February-2014
* @author Tyler Wible
* @since 31-July-2013
*/
public class createSWAT_MODFLOW_files {
  public static void main(String[] args) throws IOException {

    //Ask user for inputs
    String directory = fetchDirectory();
    int num_Grids = fetchGrids();
    int num_HRUs = fetchHRUs();
    int num_DHRUs = fetchDHRUs();
    String mfRiverFile = fetchMFriverFile();
    int num_RiverGrids = fetchRiverGrids();

    //Set inputs
    String[] inputs = new String[3];
    inputs[0] = directory;


    //Create grid2dhru file
    inputs[1] = String.valueOf(num_Grids);
    inputs[2] = String.valueOf(num_DHRUs);
    map_grid2dhru.main(inputs);


    //Create dhru2grid file
    inputs[1] = String.valueOf(num_Grids);
    inputs[2] = String.valueOf(num_DHRUs);
    map_dhru2grid.main(inputs);


    //Create dhru2hru file
    inputs[1] = String.valueOf(num_HRUs);
    inputs[2] = String.valueOf(num_DHRUs);
    map_dhru2hru.main(inputs);


    //Create river2grid file
    inputs[1] = mfRiverFile;
    inputs[2] = String.valueOf(num_RiverGrids);
    map_river2grid.main(inputs);
  }
  static String fetchDirectory(){
    String fileString = JOptionPane.showInputDialog("Please enter the file path to the
directory where the input files are located. Ex: 'C:/myFolder/myOtherFolder'", "Enter file
directory");
    return fileString;
  }
  static int fetchGrids(){
    String gridString = JOptionPane.showInputDialog("Please enter the total number
(including inactive cells) of MODFLOW grid cells in the MODFLOW model", "Enter
Number of Grid Cells");
    int gridInt = Integer.parseInt(gridString);
    return gridInt;
  }
  static int fetchHRUs(){
    String hruString = JOptionPane.showInputDialog("Please enter the total number of
SWAT Hydrologic Response Units (HRUs) in the SWAT model", "Enter Number of
HRUs");
    int hruInt = Integer.parseInt(hruString);
    return hruInt;
  }
  static int fetchDHRUs(){
    String dhruString = JOptionPane.showInputDialog("Please enter the total number of
Disaggregated-Hydrologic Response Units (DHRUs) in the SWAT/MODFLOW/UZF-
RT3D model", "Enter Number of DHRUs");
    int dhruInt = Integer.parseInt(dhruString);
    return dhruInt;
  }
  static String fetchMFriverFile(){
    String fileString = JOptionPane.showInputDialog("Please enter the name of the river
pakage file in the MODFLOW model (ex: 'myMODFLOWFile.riv')", "Enter
MODFLOW river file name");
    return fileString;
  }
  static int fetchRiverGrids(){
    String dhruString = JOptionPane.showInputDialog("Please enter the total number of
MODFLOW river grid cells in the MODFLOW model", "Enter Number of river grid
cells");
    int dhruInt = Integer.parseInt(dhruString);
    return dhruInt;
  }
}
```

MAP_DHRU2GRID.JAVA

```
package SWAT_MODFLOW;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Formatter;
import java.util.Scanner;

/**
* Last Updated: 10-March-2014
* @author Tyler Wible
* @since 11-April-2013
*/
public class map_dhru2grid {
  public static void main(String[] args) throws IOException {
    //Inputs
    String directory = args[0];
    int num_Grids = Integer.parseInt(args[1]);
    int num_DHRUs = Integer.parseInt(args[2]);

    System.out.println("Creating map_dhru2grid.txt...");
    readFile(directory, num_Grids, num_DHRUs);
    System.out.println("Done");
  }
  /**
   * Reads the attribute table file MOD_HRU.csv containing the grid IDs, HRU IDs, their
respective areas and their area of overlap
   * @param directory  the path to the input file (ex. C:/temp)
   * @param numGrids  the total number of grid cells (active and inactive) in the
MODFLOW model
   * @param num_DHRUs  the total number of disaggregated HRUs in the SWAT model
   * @throws IOException
   */
  public static void readFile(String directory, int numGrids, int num_DHRUs) throws
IOException {
    // read the file
    String path = directory + "/DHRU_grid.csv";
    File file = new File(path);
    Scanner scanner = new Scanner(file);

    // Read header
    String[] header = scanner.nextLine().split(",");

    //Get indices for the desired columns
    int index_grid_id = 0;
    int index_grid_area = 0;
    int index_dhru_id = 0;
    int index_overlap_area = 0;
    for(int i=0; i<header.length; i++){
      header[i] = header[i].replace('"', '%');
      header[i] = header[i].replaceAll("%", "");
      if(header[i].equalsIgnoreCase("grid_id")){
        index_grid_id = i;
      }else if(header[i].equalsIgnoreCase("grid_area")){
        index_grid_area = i;
      }else if(header[i].equalsIgnoreCase("dhru_id")){
        index_dhru_id = i;
      }else if(header[i].equalsIgnoreCase("overlap_a")){
        index_overlap_area = i;
      }
    }

    //Check for missing information
    if(index_grid_id == 0){
      throw new IOException("Error encountered, there is no attribute column 'grid_id' in
DHRU_grid.csv");
    }
    if(index_grid_area == 0){
      throw new IOException("Error encountered, there is no attribute column 'grid_area'
in DHRU_grid.csv");
    }
    if(index_dhru_id == 0){
      throw new IOException("Error encountered, there is no attribute column 'dhru_id' in
DHRU_grid.csv");
    }
    if(index_overlap_area == 0){
      throw new IOException("Error encountered, there is no attribute column 'overlap_a'
in DHRU_grid.csv");
    }
    // Read and parse text file
    ArrayList<String> DHRU_Num = new ArrayList<String>();          // DHRU Num
    ArrayList<Double> percent_Area_GRID = new ArrayList<Double>(); // area DHRU
inside given grid / total area grid
    ArrayList<String> grid_Num = new ArrayList<String>();        // Grid number
    while(scanner.hasNextLine()){
      String line = scanner.nextLine();

      // Split the read data
      String[] data_Splitted = line.split(",");

      // Grab values from text file
      double total_area_of_GRID = Double.parseDouble(data_Splitted[index_grid_area]);
      double area_of_HRU_within_GRID =
Double.parseDouble(data_Splitted[index_overlap_area]);

      //Calculate percent area of HRU within grid
      double percent_area_of_GRID_Double = area_of_HRU_within_GRID /
total_area_of_GRID;

      //Save values from text file
      DHRU_Num.add(data_Splitted[index_dhru_id]);
      percent_Area_GRID.add(percent_area_of_GRID_Double);
      grid_Num.add(data_Splitted[index_grid_id]);
    }
    scanner.close();

    write_modified_HRU_MOD(directory, DHRU_Num, percent_Area_GRID, grid_Num,
numGrids, num_DHRUs);
  }
  /**
   * Takes the list of grid/hru ids and the percent area of HRUs contributing to a given
grid and creates the SM linkage file
   * map_hru2grid.txt
   * @param directory  the path to the output file (ex. C:/temp)
   * @param DHRU_Num list of HRU ID numbers
   * @param percent_Area_GRID list of percent grid within the above HRU ID
   * @param grid_Num  list of grid ID numbers
   * @param numGrids  the total number of grid cells (active and inactive) in the
MODFLOW model
   * @param num_DHRUs  the total number of disaggregated HRUs in the SWAT model
   * @throws IOException
   */
  public static void write_modified_HRU_MOD(String directory,
                    ArrayList<String> DHRU_Num,
                    ArrayList<Double> percent_Area_GRID,
                  ArrayList<String> grid_Num,
                  int numGrids,
                  int num_DHRUs) throws IOException {

    String path = directory + "/map_dhru2grid.txt";
    File file = new File(path);
    boolean fileTF = file.isFile();
    if(fileTF){
      boolean fileDeleteTF = file.delete();
      if(!fileDeleteTF){
        System.out.println("The file (" + path + ") could not be deleted");
      }
    }

    ArrayList<ArrayList<String>> gridIndexList = new ArrayList<ArrayList<String>>();
    int maxSize = 0;
    for(int j=1; j <= numGrids; j++){ // iterating over grid number
      System.out.println(j);
      // Find all positions containing current Grid number
      ArrayList<String> DHRUindex_for_currentGrid = new ArrayList<String>(); //
arraylist containing all HRUs within current Grid Num"
      for(int i = 0; i < grid_Num.size(); i++){
        if(j == (int) Double.parseDouble(grid_Num.get(i))){
          // Gives me an array containing all positions of HRUs which contribute to current
grid
          DHRUindex_for_currentGrid.add(String.valueOf(i));
        }
      }
      if(DHRUindex_for_currentGrid.size() > maxSize) maxSize =
DHRUindex_for_currentGrid.size();
      gridIndexList.add(DHRUindex_for_currentGrid);
    }

    // Format for which the text file will be written
    Formatter HRU_MOD = new Formatter(new FileWriter(file, true));
    // Write out the total number of MODFLOW grid cells which intersect with the SWAT
hrus and what the maximum number of SWAT hrus is that intersect this
    HRU_MOD.format("%1$12s%2$12s%n", numGrids, maxSize);

    for(int j=1; j <= numGrids; j++){ // iterating over grid number
      ArrayList<String> currentGrid_DHRUindexList = gridIndexList.get(j-1);

      // format the grid number as first column and the number of HRUs contributing to this
grid as the second column
      HRU_MOD.format("%1$12s%2$12s%n", j, currentGrid_DHRUindexList.size());
      //Print ID of HRUs contributing to the current grid
      for(int i = 0; i < currentGrid_DHRUindexList.size(); i++){
        int hruIndex = Integer.parseInt(currentGrid_DHRUindexList.get(i));
        int hruNumber = (int) Double.parseDouble(DHRU_Num.get(hruIndex));
        HRU_MOD.format("%1$12s", hruNumber);
      }
      HRU_MOD.format("%n", "");


      //Print Percent areas of each HRU contributing to the current grid
      for(int i = 0; i < currentGrid_DHRUindexList.size(); i++){
        int hruIndex = Integer.parseInt(currentGrid_DHRUindexList.get(i));
        HRU_MOD.format("%1$12.5f", percent_Area_GRID.get(hruIndex));
      }
      HRU_MOD.format("%n", "");
    }
    gridIndexList.clear();
    HRU_MOD.close();
  }
}
//Format of output
//390      59   (total number of grids to be read in) (the maximum number of dhrus that
contribute to a single grid)
//1        4    (grid#)    (# of dhrus contributing to this grid)
//hru2_ID   hru12_ID   hru31_ID   hru37_ID  (list of dhru id #s contributing to grid#)
//0.36      0.25       0.04       0.35      (list of %areas of dhru contributing to grid#, adds to 1)
```

MAP_DHRU2HRU.JAVA

```
package SWAT_MODFLOW;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Formatter;
import java.util.Scanner;


/**
* Last Updated: 10-March-2014
* @author Tyler Wible
* @since 11-June-2013
*/
public class map_dhru2hru {
  public static void main(String[] args) throws IOException {
    //Inputs
    String directory = args[0];
    int num_HRUs = Integer.parseInt(args[1]);
    int num_DHRUs = Integer.parseInt(args[2]);

    System.out.println("Creating map_dhru2hru.txt...");
    readFile(directory, num_DHRUs, num_HRUs);
    System.out.println("Done");
  }
  /**
   * Reads the attribute table file MOD_HRU.csv containing the grid IDs, HRU IDs, their
respetive areas and their area of overlap
   * @param directory  the path to the input file (ex. C:/temp)
   * @param num_DHRUs  the total number of disaggregated HRUs in the SWAT model
   * @param num_HRUs  the total number of aggregated (normal SWAT) HRUs in the
SWAT models
   * @throws IOException
   */
  public static void readFile(String directory, int num_DHRUs, int num_HRUs) throws
IOException {
    // read the file
    String path = directory + "/FullDHRU.csv";
    File file = new File(path);
    Scanner scanner = new Scanner(file);

    // Read header
    String[] header = scanner.nextLine().split(",");

    //Get indices for the desired columns
    int index_hru_id = 0;
    int index_hru_area = 0;
    int index_sub_id = 0;
    int index_dhru_id = 0;
    int index_dhru_area = 0;
    for(int i=0; i<header.length; i++){
      header[i] = header[i].replace('"', '%');
      header[i] = header[i].replaceAll("%", "");
      if(header[i].equalsIgnoreCase("hru_id")){
        index_hru_id = i;
      }else if(header[i].equalsIgnoreCase("Shape_Area")){
        index_hru_area = i;
      }else if(header[i].equalsIgnoreCase("SUBBASIN")){
        index_sub_id = i;
      }else if(header[i].equalsIgnoreCase("dhru_id")){
        index_dhru_id = i;
      }else if(header[i].equalsIgnoreCase("dhru_area")){
        index_dhru_area = i;
      }
    }

    //Check for missing information
    if(index_hru_id == 0){
      throw new IOException("Error encountered, there is no attribute column 'hru_id' in
FullDHRU_intersect.csv");
    }
    if(index_hru_area == 0){
      throw new IOException("Error encountered, there is no attribute column
'Shape_Area' in FullDHRU_intersect.csv");
    }
    if(index_sub_id == 0){
      throw new IOException("Error encountered, there is no attribute column
'SUBBASIN' in FullDHRU_intersect.csv");
    }
    if(index_dhru_id == 0){
      throw new IOException("Error encountered, there is no attribute column 'dhru_id' in
FullDHRU_intersect.csv");
    }
    if(index_dhru_area == 0){
      throw new IOException("Error encountered, there is no attribute column 'dhru_area'
in FullDHRU_intersect.csv");
    }


    // Read and parse now reduced text file (should have 1 line per dhru)
    ArrayList<String> HRU_Num = new ArrayList<String>();          //HRU Num
    ArrayList<Double> percent_Area_HRU = new ArrayList<Double>();  //area DHRU
inside given HRU / total area HRU
    ArrayList<String> dhru_Num = new ArrayList<String>();         //DHRU number
    ArrayList<Integer> subbasin_id = new ArrayList<Integer>();     //subbasin id number
containing the current river ID number

    while(scanner.hasNextLine()){
      // Split the read data
      String[] data_Splitted = scanner.nextLine().split(",");

      // Grab values from text file
      double total_area_of_HRU = Double.parseDouble(data_Splitted[index_hru_area]);
      double area_of_Dhru_within_HRU =
Double.parseDouble(data_Splitted[index_dhru_area]);

      //Calculate percent area of dhru within HRU
      double percent_area_of_HRU_Double = area_of_Dhru_within_HRU /
total_area_of_HRU;

      //Save values from text file
      HRU_Num.add(data_Splitted[index_hru_id]);
      percent_Area_HRU.add(percent_area_of_HRU_Double);
      dhru_Num.add(data_Splitted[index_dhru_id]);
      subbasin_id.add(Integer.parseInt(data_Splitted[index_sub_id]));
    }
    scanner.close();

    write_map_grid2hru(directory, HRU_Num, percent_Area_HRU, dhru_Num,
subbasin_id, num_DHRUs, num_HRUs);
  }
  /**
   * Takes the list of dhru/hru ids and the percent area of dhrus contributing to a given
HRU and creates the SM linkage file
   * map_dhru2hru.txt
   * @param directory  the path to the output file (ex. C:/temp)
   * @param HRU_Num list of HRU ID numbers
   * @param percent_Area_hru list of percent dhru within the above HRU ID
   * @param dhru_Num  list of dhru ID numbers
   * @param subbasin_id  list of subbasin ID numbers
   * @param num_DHRUs  the total number of disaggregated HRUs in the SWAT model
   * @param num_HRUs  the total number of aggregated (normal SWAT) HRUs in the
SWAT model
   * @throws IOException
   */
  public static void write_map_grid2hru(String directory,
                  ArrayList<String> HRU_Num,
                  ArrayList<Double> percent_Area_hru,
                  ArrayList<String> dhru_Num,
                  ArrayList<Integer> subbasin_id,
                  int num_DHRUs,
                  int num_HRUs)throws IOException {

    String path = directory + "/map_dhru2hru.txt";
    File file = new File(path);
    boolean fileTF = file.isFile();
    if(fileTF){
      boolean fileDeleteTF = file.delete();
      if(!fileDeleteTF){
        System.out.println("The file (" + path + ") could not be deleted");
      }
    }
   ArrayList<ArrayList<String>> hruIndexList = new ArrayList<ArrayList<String>>();
   int maxSize = 0;
   for(int j=1; j <= num_HRUs; j++){ // iterating over hru number
     System.out.println(j);
     // Find all positions containing current hru number
     ArrayList<String> DHRUindex_for_currentHRU = new ArrayList<String>(); //
arraylist containing all dhrus within current HRU
     for(int i = 0; i < HRU_Num.size(); i++){
       if(j == (int) Double.parseDouble(HRU_Num.get(i))){
         // Gives an array containing all the indexes of dhrus which contribute to current
HRU
         DHRUindex_for_currentHRU.add(String.valueOf(i));
       }
     }
     if(DHRUindex_for_currentHRU.size() > maxSize) maxSize =
DHRUindex_for_currentHRU.size();
     hruIndexList.add(DHRUindex_for_currentHRU);
   }

   // Format for which the text file will be written
   Formatter outputFile = new Formatter(new FileWriter(file, true));
   // Write out the total number of disaggregated HRUs contained in this file
   outputFile.format("%1$12s%2$12s%n", num_HRUs, maxSize);

   for(int j=1; j <= num_HRUs; j++){ // iterating over hru number
     ArrayList<String> currentHRU_DHRUindexList = hruIndexList.get(j-1);

     // format the HRU number as first column and the number of dhrus contributing to
this HRU as the second column
     int index = Integer.parseInt(currentHRU_DHRUindexList.get(0));
     int sub_id = subbasin_id.get(index);
     outputFile.format("%1$12s%2$12s%3$12s%n", j,
currentHRU_DHRUindexList.size(), sub_id);

     //Print the dhru ids for the grids contributing to the current dhru
     for(int i = 0; i < currentHRU_DHRUindexList.size(); i++){
       int gridIndex = Integer.parseInt(currentHRU_DHRUindexList.get(i));
       int rowNumber = (int) Double.parseDouble(dhru_Num.get(gridIndex));
       outputFile.format("%1$12s", rowNumber);
     }
     outputFile.format("%n", "");

     //Print Percent areas of each grid contributing to the current dhru
     for(int i = 0; i < currentHRU_DHRUindexList.size(); i++){
       int gridIndex = Integer.parseInt(currentHRU_DHRUindexList.get(i));
       outputFile.format("%1$12.5f", percent_Area_hru.get(gridIndex));
```

```
      }
      outputFile.format("%n", "");
    }
    outputFile.close();//tcw
  }
}
//Format of output:
//390  (total number of disaggregated HRUs to be read in)
//1     4     1    (aggregated hru#)  (# of dhrus contributing to this hru) (subbasinID)
//dhru1_ID  dhru12_ID  dhru31_ID  dhru37_ID  (list of dhru id #s contributing to hru#)
//0.36      0.25       0.04       0.35      (list of %areas of dhru contributing to hru#, adds to 1)
```

MAP_GRID2DHRU.JAVA

```
package SWAT_MODFLOW;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Formatter;
import java.util.Scanner;

/**
* Last Updated: 10-March-2014
* @author Tyler Wible
* @since 11-June-2013
*/
public class map_grid2dhru {
  public static void main(String[] args) throws IOException {
    //Inputs
    String directory = args[0];
    int num_Grids = Integer.parseInt(args[1]);
    int num_DHRUs = Integer.parseInt(args[2]);

    System.out.println("Creating map_grid2dhru.txt...");
    readFile(directory, num_Grids, num_DHRUs);
    System.out.println("Done");
  }
  /**
   * Reads the attribute table file MOD_HRU.csv containing the grid IDs, HRU IDs, their
respective areas and their area of overlap
   * @param directory  the path to the input file (ex. C:/temp)
   * @param numGrids  the total number of grid cells (active and inactive) in the
MODFLOW model
   * @param numDHRUs  the total number of disaggregated HRUs in the SWAT model
   * @throws IOException
   */
  public static void readFile(String directory, int numGrids, int numDHRUs) throws
IOException {
    // read the file
    String path = directory + "/DHRU_grid.csv";
    File file = new File(path);
    Scanner scanner = new Scanner(file);

    // Read header
    String[] header = scanner.nextLine().split(",");

    //Get indices for the desired columns
    int index_col_row = 0;
    int index_grid_id = 0;
    int index_dhru_id = 0;
    int index_dhru_area = 0;
    int index_overlap_area = 0;
    for(int i=0; i<header.length; i++){
      header[i] = header[i].replace("", '%');
      header[i] = header[i].replaceAll("%", "");
      if(header[i].equalsIgnoreCase("Col_Row")){
        index_col_row = i;
      }else if(header[i].equalsIgnoreCase("grid_id")){
        index_grid_id = i;
      }else if(header[i].equalsIgnoreCase("dhru_id")){
        index_dhru_id = i;
      }else if(header[i].equalsIgnoreCase("dhru_area")){
        index_dhru_area = i;
      }else if(header[i].equalsIgnoreCase("overlap_a")){
        index_overlap_area = i;
      }
    }

    //Check for missing information
    if(index_col_row == 0){
      throw new IOException("Error encountered, there is no attribute column 'Col_Row'
in DHRU_grid.csv");
    }
    if(index_grid_id == 0){
      throw new IOException("Error encountered, there is no attribute column 'grid_id' in
DHRU_grid.csv");
    }
    if(index_dhru_id == 0){
      throw new IOException("Error encountered, there is no attribute column 'dhru_id' in
DHRU_grid.csv");
```

```
    }
    if(index_dhru_area == 0){
      throw new IOException("Error encountered, there is no attribute column 'dhru_area'
in DHRU_grid.csv");
    }
    if(index_overlap_area == 0){
      throw new IOException("Error encountered, there is no attribute column 'overlap_a'
in DHRU_grid.csv");
    }

    // Read and parse text file
    ArrayList<String> DHRU_Num = new ArrayList<String>();          // HRU Num
    ArrayList<Double> percent_Area_HRU = new ArrayList<Double>();  // area grid
inside given HRU / total area HRU
    ArrayList<String> grid_Num = new ArrayList<String>();          // Grid number
    ArrayList<String> grid_row = new ArrayList<String>();          // Grid row id
    ArrayList<String> grid_col = new ArrayList<String>();          // Grid column id
    while(scanner.hasNextLine()){
      String line = scanner.nextLine();

      // Split the read data
      String[] data_Splitted = line.split(",");

      // Grab values from text file
      double total_area_of_DHRU =
Double.parseDouble(data_Splitted[index_dhru_area]);
      double area_of_GRID_within_DHRU =
Double.parseDouble(data_Splitted[index_overlap_area]);

      //Calculate percent area of grid within HRU
      double percent_area_of_DHRU_Double = area_of_GRID_within_DHRU /
total_area_of_DHRU;

      //Save values from text file
      DHRU_Num.add(data_Splitted[index_dhru_id]);
      percent_Area_HRU.add(percent_area_of_DHRU_Double);
      grid_Num.add(data_Splitted[index_grid_id]);
      grid_row.add(columnRow(data_Splitted[index_col_row], false)); //grid row#
      grid_col.add(columnRow(data_Splitted[index_col_row], true));  //grid column#
    }
    scanner.close();

    write_map_grid2dhru(directory, DHRU_Num, percent_Area_HRU, grid_Num,
grid_row, grid_col, numGrids, numDHRUs);
  }
  /**
```

```
   * Converts the column - row string (ex. "3 - 49" is column 3 row 49) into either its
column or row index only
   * @param column_row  the "column - row" string (ex. "3 - 49")
   * @param columnTrue  if true, this returns the column index (ex. "3") if false it returns
the row index (ex. "49")
   * @return  The column or row index dependiong on the value of columnTrue
   */
  public static String columnRow(String column_row, boolean columnTrue){
    //Convers the strin Col -  row into

    String[] array = column_row.substring(1, column_row.length() - 1).split("-");
    String column = array[0].trim();
    String row = array[1].trim();

    String returnValue = row;
    if(columnTrue){
      returnValue = column;
    }

    return returnValue;
  }
  /**
   * Takes the list of grid/hru ids and the percent area of grids contributing to a given
HRU and creates the SM linkage file
   * map_grid2hru.txt
   * @param directory  the path to the output file (ex. C:/temp)
   * @param DHRU_Num list of HRU ID numbers
   * @param percent_Area_HRU list of percent grid within the above HRU ID
   * @param grid_Num  list of grid ID numbers
   * @param grid_row  list of grid row numbers
   * @param grid_col  list of grid column numbers
   * @param numGrids  the total number of grid cells (active and inactive) in the
MODFLOW model
   * @param numDHRUs  the total number of disaggregated HRUs in the SWAT model
   * @throws IOException
   */
  public static void write_map_grid2dhru(String directory,
                    ArrayList<String> DHRU_Num,
                    ArrayList<Double> percent_Area_HRU,
                    ArrayList<String> grid_Num, //unused
                    ArrayList<String> grid_row,
                    ArrayList<String> grid_col,
                    int numGrids, //unused
                    int numDHRUs)throws IOException {

    String path = directory + "/map_grid2dhru.txt";
    File file = new File(path);
    boolean fileTF = file.isFile();
    if(fileTF){
      boolean fileDeleteTF = file.delete();
      if(!fileDeleteTF){
        System.out.println("The file (" + path + ") could not be deleted");
      }
    }

    ArrayList<ArrayList<String>> dhruIndexList = new ArrayList<ArrayList<String>>();
    int maxSize = 0;
    for(int j=1; j <= numDHRUs; j++){ // iterating over dhru number
      System.out.println(j);
      // Find all positions containing current Grid number
      ArrayList<String> gridIndex_for_currentDHRU = new ArrayList<String>(); //
arraylist containing all grids within current HRU
      for(int i = 0; i < DHRU_Num.size(); i++){
        if(j == (int) Double.parseDouble(DHRU_Num.get(i))){
          // Gives an array containing all the indexes of grids which contribute to current
HRU
          gridIndex_for_currentDHRU.add(String.valueOf(i));
        }
      }
      if(gridIndex_for_currentDHRU.size() > maxSize) maxSize =
gridIndex_for_currentDHRU.size();
      dhruIndexList.add(gridIndex_for_currentDHRU);
    }

    // Format for which the text file will be written
    Formatter MOD_HRU = new Formatter(new FileWriter(file, true));
    // Write out the total number of dissaggregated hrus which intersect with the
MODFLOW grid cells and the maximum number of MODFLOW grid cells that intersect
a single dhru
    MOD_HRU.format("%1$12s%2$12s%n", numDHRUs, maxSize);

    for(int j=1; j <= numDHRUs; j++){ // iterating over dhru number
      ArrayList<String> currentDHRU_gridIndexList = dhruIndexList.get(j-1);

      // format the dHRU number as first column and the number of grids contributing to
this HRU as the second column
      MOD_HRU.format("%1$12s%2$12s%n", j, currentDHRU_gridIndexList.size());

      //Print the row ids for the grids contributing to the current dhru
      for(int i = 0; i < currentDHRU_gridIndexList.size(); i++){
        int gridIndex = Integer.parseInt(currentDHRU_gridIndexList.get(i));
        int rowNumber = (int) Double.parseDouble(grid_row.get(gridIndex));
      MOD_HRU.format("%1$12s", rowNumber);
    }
    MOD_HRU.format("%n", "");

    //Print the column ids for the grids contributing to the current dhru
    for(int i = 0; i < currentDHRU_gridIndexList.size(); i++){
      int gridIndex = Integer.parseInt(currentDHRU_gridIndexList.get(i));
      int columnNumber = (int) Double.parseDouble(grid_col.get(gridIndex));
      MOD_HRU.format("%1$12s", columnNumber);
    }
    MOD_HRU.format("%n", "");


    //Print Percent areas of each grid contributing to the current dhru
    for(int i = 0; i < currentDHRU_gridIndexList.size(); i++){
      int gridIndex = Integer.parseInt(currentDHRU_gridIndexList.get(i));
      MOD_HRU.format("%1$12.5f", percent_Area_HRU.get(gridIndex));
    }
    MOD_HRU.format("%n", "");
  }
  MOD_HRU.close();//tcw
 }
}
//Format of outout: tcw
//390        62   (total number of dHRUs to be read in)    (the maximum number of grids
that contribute to a single dhru)
//1          4    (hru#)    (# of grids contributing to this dhru)
//grid2_row_ID  grid12_row_ID  grid31_row_ID  grud37_row_ID  (list of grid row id #s
contributing to dhru#)
//grid2_col_ID  grid12_col_ID  grid31_col_ID  grud37_col_ID  (list of grid column id #s
contributing to dhru#)
//0.36       0.25       0.04       0.35       (list of %areas of grid contributing to
dhru#, adds to 1)
```

MAP_RIVER2GRID.JAVA

```
package SWAT_MODFLOW;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Formatter;
import java.util.Scanner;
```

```
/**
* Last Updated: 28-February-2014
* @author Tyler Wible
* @since 18-April-2013
*/
public class map_river2grid {
  public static void main(String[] args) throws IOException {
    //Inputs
    String directory = args[0];
    String mfRiverFile = args[1];
    int num_RiverGrids = Integer.parseInt(args[2]);

    System.out.println("Creating map_river2grid.txt...");
    writeResults(directory, mfRiverFile, num_RiverGrids);
    System.out.println("Done");
  }
  /**
   * Converts the column - row string (ex. "3 - 49" is column 3 row 49) into either its
column or row index only
   * @param column_row  the "column - row" string (ex. "3 - 49")
   * @param columnTrue  if true, this returns the column index (ex. "3") if false it returns
the row index (ex. "49")
   * @return  The column or row index depending on the value of columnTrue
   */
  public static String columnRow(String column_row, boolean columnTrue){
    //Convers the strin Col -  row into

    String[] array = column_row.substring(1, column_row.length() - 1).split("-");
    String column = array[0].trim();
    String row = array[1].trim();

    String returnValue = row;
    if(columnTrue){
      returnValue = column;
    }

    return returnValue;
  }
  /**
   * Reads the attribute table file as a CSV file containing information on the river
   * segments in SWAT intersected with the MODFLOW grid and pull out information on
grid
   * cell IDs and the subbasins of the river
   * @param fileLocation  the path to the input file (ex. C:/temp)
   * @return  selected contents of the CSV as a String[][] array
   * @throws IOException
   */
  public static String[][] readRiverGridFile(String fileLocation) throws IOException{
    //Open a reader for the results file
    String path = fileLocation + "/rivergrid.csv";
    FileReader file_to_read = new FileReader(path);
    BufferedReader reader = new BufferedReader(file_to_read);

    //Read out some of the contents of the input file
    ArrayList<String> data = new ArrayList<String>();
    String currentLine = "";
    while((currentLine = reader.readLine()) !=null){
      data.add(currentLine);
    }
    reader.close();
    file_to_read.close();

    //Populate data
    String[][] resultArray = new String[data.size()-1][4];
    int index_col_row = 0;
    int index_riv_id = 0;
    int index_sub_id = 0;
    int index_riv_len = 0;
    for(int i=0; i<data.size(); i++){//Loop rows of data
      String[] currentRow = data.get(i).split(",");

      if(i == 0){
        for(int j=0; j<currentRow.length; j++){
          currentRow[j] = currentRow[j].replace('"', '%');
          currentRow[j] = currentRow[j].replaceAll("%", "");
          if(currentRow[j].equalsIgnoreCase("Col_Row")){
            index_col_row = j;
          }else if(currentRow[j].equalsIgnoreCase("riv_id")){
            index_riv_id = j;
          }else if(currentRow[j].equalsIgnoreCase("Subbasin")){
            index_sub_id = j;
          }else if(currentRow[j].equalsIgnoreCase("rgrid_len")){
            index_riv_len = j;
          }
        }

        //Check for missing information
        if(index_col_row == 0){
          throw new IOException("Error encountered, there is no attribute column
'Col_Row' in rivergrid.csv");
        }
        if(index_riv_id == 0){
          throw new IOException("Error encountered, there is no attribute column 'riv_id' in
rivergrid.csv");
        }
        if(index_sub_id == 0){
          throw new IOException("Error encountered, there is no attribute column 'Subbasin'
in rivergrid.csv");
        }
        if(index_riv_len == 0){
          throw new IOException("Error encountered, there is no attribute column 'rgrid_len'
in rivergrid.csv");
        }


      }else{
        //Keep desired information
        resultArray[i-1][0] = currentRow[index_riv_id];                    //riverID#
        resultArray[i-1][1] = currentRow[index_riv_len];                   //lengthOfRiver in
grid#
        resultArray[i-1][2] = columnRow(currentRow[index_col_row], true);  //grid
column#
        resultArray[i-1][3] = columnRow(currentRow[index_col_row], false); //grid row#
      }
    }

    return resultArray;
  }
  /**
   * read the MODLFOW river package file to get the order/info. about the river grid cells
so the right
   * SWAT river information is passed to the correct MODFLOW river grid cell
   * @param filePath  the full file path to the river package file (ex.
C:/tempFolder/myRivepackage.txt)
   * @param num_RiverGrids  the total number of MODFLOW river grids in the model
   * @return  selected contents of the river package as a String[][] array
   * @throws IOException
   */
  public static int[][] readMODFLOWrivFile(String filePath, int num_RiverGrids) throws
IOException{
    // intialize scanner (reader)
    File file = new File(filePath);
    Scanner scanner = new Scanner(file);

    //Read in header information information
    boolean readNextLine = true;
    while(readNextLine){
      String currentLine = scanner.nextLine();
      try{
        if(currentLine.substring(0,1).equals("#")){
          readNextLine = true;
        }else{
          currentLine = scanner.nextLine();
          readNextLine = false;
        }
      }catch(IndexOutOfBoundsException e){
        currentLine = scanner.nextLine();
        readNextLine = false;
      }
    }


    //Populate data
    int[][] resultArray = new int[num_RiverGrids][3];
    for(int i=1; i<=num_RiverGrids; i++){//Loop for number of river cells
      String[] columns = scanner.nextLine().split("\\s+");

      //Keep desired information
      resultArray[i-1][0] = i;                          //river grid #
      resultArray[i-1][1] = Integer.parseInt(columns[2]); //grid row#
      resultArray[i-1][2] = Integer.parseInt(columns[3]); //grid column#
    }

    return resultArray;
  }
  /**
   *
   * @param fileLocation
   * @param subbasinMax
   * @throws IOException
   */
  public static void writeResults(String fileLocation, String mfRiverFile, int
num_RiverGrids) throws IOException{

    //Read River/grid Attribute table file
    String[][] riverGridData = readRiverGridFile(fileLocation);

    //Read MODFLOW .riv file
    int[][] mfRiverData = readMODFLOWrivFile(fileLocation + "/" + mfRiverFile,
num_RiverGrids);

    //Delete the old map_rive2grid.txt if it exists
    String path = fileLocation + "/map_river2grid.txt";
    File file = new File(path);
    boolean fileTF = file.isFile();
    if(fileTF){
      boolean fileDeleteTF = file.delete();
      if(!fileDeleteTF){
        System.out.println("The file (" + path + ") could not be deleted");
      }
    }

    // Format for which the text file will be written
    Formatter riverFile = new Formatter(new FileWriter(file, true));
    riverFile.format("%1$12s%n", num_RiverGrids);//Write the total number of
MODFLOW grids for dimensioning purposes


    for(int j=1; j <= num_RiverGrids; j++){ // iterating over river grid number
      System.out.println(j);
      int mfRow = mfRiverData[j-1][1];
      int mfColumn = mfRiverData[j-1][2];

      // Find all positions containing current grid number
      ArrayList<Integer> index_for_river_Num_for_Current_grid_Num = new
ArrayList<Integer>(); // arraylist containing all river within current grid number

      for(int i = 0; i < riverGridData.length; i++){
        int row = (int) Double.parseDouble(riverGridData[i][3]);
        int column  = (int) Double.parseDouble(riverGridData[i][2]);

        if(mfRow == row && mfColumn == column){
          // Gives me an array containing all positions of rivers which contribute to current
grid
          index_for_river_Num_for_Current_grid_Num.add(i);
        }
      }
      riverFile.format("%1$12s%2$12s%n", j,
index_for_river_Num_for_Current_grid_Num.size()); // format the grid number as first
column and the number of rivers contributing to this as the second column


      //Print ID of rivers contributing to the current grid
      for(int i = 0; i < index_for_river_Num_for_Current_grid_Num.size(); i++){
        int riverIndex = index_for_river_Num_for_Current_grid_Num.get(i);
        int riverNumber = (int) Double.parseDouble(riverGridData[riverIndex][0]);
        riverFile.format("%1$12s", riverNumber);
      }
      riverFile.format("%n", "");
      //Print length of river within current grid
      for(int i = 0; i < index_for_river_Num_for_Current_grid_Num.size(); i++){
        int riverIndex = index_for_river_Num_for_Current_grid_Num.get(i);
        double riverLength = Double.parseDouble(riverGridData[riverIndex][1]);
        riverFile.format("%1$12.5f", riverLength);
      }
      riverFile.format("%n", "");
      index_for_river_Num_for_Current_grid_Num.clear();
    }
    riverFile.close();//tcw
  }
}
//Format of outout: tcw
//
//57       (total # of  river cells in MODFLOW)
//36.2      591.25      299.04      350.12     (list of thickness of lengths of river segments of
grid cell)
//1        4    (grid#)    (# of river segments contributing to this grid)
//rivr2_ID  rivr12_ID   rivr31_ID   rivr37_ID  (list of river id #s which are in this grid
cell)
//36.2      591.25      299.04      350.12     (list of lengths of river segment in this grid cell)
```

APPENDIX III: EXAMPLE SWAT-MODFLOW LINKAGE INPUT FILES

GENERAL FORMATTING NOTES

- These files are space separated, aka the number of spaces between each value does not matter (i.e. “1   2” is read in the same as “1 2”).
- The function of the file is included in the name, the first element is the current format of a variable in the model; the second element is the desired/converted output of that variable for use in the model.
  - For example with, map_dhru2grid.txt
  - The first element is “dhru” meaning data is in a format of an array with an element per spatially disaggregated HRU.
  - The second element, “grid,” is the desired conversion of the variable to a 2D array of grid rows and columns, a format used by MODFLOW.
  - Therefore this file contains information of which dhrus contribute to each grid cell listed in this file to allow conversion of variables in this fashion.

EXPLANATION OF: “map_dhru2grid.txt”

This file contains an entry for every MODFLOW grid cell ID (inactive or active) as documented in Appendix I to create. The file starts with the first grid cell ID and ends with last grid cell ID (i.e. 1 to number of grid cells). Listed below are terms used in the next section to label the content of the file.

- tLines = number of grids in the MODFLOW model to read in
- DHRU = Disaggregated Hydrologic Response Unit

- A-HRU = the percent area of an DHRU inside a given grid cell # (Area DHRU inside grid # / Area of grid #)
- Grid # = the grid cell ID number for the MODFLOW model
- maxContr = the maximum number of DHRUs contributing to any grid cell used for dimensioning variables within the code

EXAMPLE FILE CONTENTS, WITH LABELS:

```
6       4                   “tLines”        “maxContr”

1       0                   “Grid#”         “# DHRUs inside Grid#”

                            “leave a blank line”

                            “leave a blank line”

2       2                   “Grid#”         “# DHRUs inside this Grid#”

17      32                  “DHRU#”         “DHRU#”

0.08    0.12                “A-HRU”         “A-HRU”

3       4                   “Grid#”         “# DHRUs inside this Grid#”

17      32      39      43      “DHRU#”     “DHRU#”     “DHRU#”     “DHRU#”

0.16    0.32    0.04    0.16    “A-HRU”     “A-HRU”     “A-HRU”     “A-HRU”
```

EXPLANATION OF EXAMPLE FILE CONTENTS

- There are a total of 6 grid cells in the MODFLOW model and the grid cell in the MODFLOW model with the most DHRUs contributing to it has 4.

- The first grid cell has 0 disaggregated HRUs which contribute to it. This is likely because the grid cell is located outside of the boundaries of the SWAT model.
    - When a grid cell has no DHRUs contributing to it, leave 2 blank lines before the next entry.
- The second grid cell has 2 DHRUs which contribute to it.
    - The first DHRU contributing to grid-ID 2 is number 17.
    - The second DHRU contributing to grid-ID 2 is number 32.
    - The first DHRU comprises 8% of the area of grid-ID 2
    - The second DHRU comprises 12% of the area of grid-ID 2
    - Because this percent area does not add up to 100% this grid cell is likely only partially covered by the SWAT model, which is allowed with this coupling.
- The third grid cell has 4 DHRUs which contribute to it.
    - The first DHRU contributing to grid-ID 3 is number 17.
    - The second DHRU contributing to grid-ID 3 is number 32.
    - The third DHRU contributing to grid-ID 3 is number 39.
    - The fourth DHRU contributing to grid-ID 3 is number 43.
    - The first DHRU comprises 16% of the area of grid-ID 3
    - The second DHRU comprises 32% of the area of grid-ID 3
    - The third DHRU comprises 4% of the area of grid-ID 3
    - The fourth DHRU comprises 16% of the area of grid-ID 3
    - Because this percent area does not add up to 100% this grid cell is likely only partially covered by the SWAT model, which is allowed with this coupling.

EXPLANATION OF: “map_dhru2hru.txt”

This file contains an entry for every SWAT HRU as created by the ArcSWAT interface. The file starts with the first HRU and ends with last HRU (i.e. 1 to number of HRUs in entire watershed). Listed below are terms used in the next section to label the content of the file.

- DHRU = Disaggregated Hydrologic Response Unit
- HRU = AHRU = Aggregated Hydrologic Response Unit (normal ArcSWAT HRUs)
- sub# = the subbasin number that the AHRU currently resides in
- A-HRU = the percent area of DHRU inside an AHRU (DHRU Area / AHRU Area)
- DHRU id = The global id of the DHRU
- maxContr = the maximum number of DHRUs within any HRU used for dimensioning variables within the code

EXAMPLE FILE CONTENTS, WITH LABELS:

```
35      4               “Total # of HRUs”   “maxContr”

1       2       1       “AHRU #”     “# DHRUs inside AHRU ID = 1”     “Sub#”

14      15              “DHRU id”    “DHRU id”

0.08    0.92            “A-HRU”      “A-HRU”

2       4       1       “AHRU #”     “# DHRUs inside AHRU ID = 2”     “Sub#”

1       3       4       7       “DHRU id”   “DHRU id”   “DHRU id”   “DHRU id”

0.08    0.12    0.4     0.4     “A-HRU”     “A-HRU”     “A-HRU”     “A-HRU”
```

EXPLANATION OF EXAMPLE FILE CONTENTS

- There are a total of 35 HRUs in the SWAT model and the HRU in the SWAT model with the most DHRUs contained within it has 4.
- The first SWAT HRU has 2 DHRUs within it and is located in sub-basin 1.
    - The first DHRU contained in HRU-1 is number 14.
    - The second DHRU contained in HRU-1 is number 15.
    - The first DHRU comprises 8% of the area of HRU-1
    - The second DHRU comprises 92% of the area of HRU-1
- The second SWAT HRU has 4 DHRUs within it and is located in sub-basin 1.
    - The first DHRU contained in HRU-2 is number 1
    - The second DHRU contained in HRU-2 is number 3
    - The third DHRU contained in HRU-2 is number 4
    - The fourth DHRU contained in HRU-2 is number 7
    - The first DHRU comprises 8% of the area of HRU-2
    - The second DHRU comprises 12% of the area of HRU-2
    - The third DHRU comprises 40% of the area of HRU-2
    - The fourth DHRU comprises 40% of the area of HRU-2

EXPLANATION OF: "map_grid2dhru.txt"

This file contains an entry for every DHRU to create as documented in Appendix I. The file starts with the first DHRU-ID and ends with last DHRU-ID (i.e. 1 to number of DHRUs). Listed below are terms used in the next section to label the content of the file.

- tLines = number of DHRUs in the coupled model to read in

- DHRU# = Disaggregated Hydrologic Response Unit ID number
- Row-ID = the row ID for the grid cell contributing to the DHRU
- Col-ID = the column ID for the grid cell contributing to the DHRU
- A-grid = the percent area of an grid cell inside a given DHRU# (Area grid cell inside DHRU# / Area of DHRU#)
- Grid # = the grid cell ID number for the MODFLOW model
- maxContr = the maximum number of grid cells contributing to any DHRU, used for dimensioning variables within the code

EXAMPLE FILE CONTENTS, WITH LABELS:

400	7	"tLines"	"maxContr"
1	0	"DHRU#"	"# grid cells inside DHRU#"
		"leave a blank line"	
		"leave a blank line"	
		"leave a blank line"	
2	1	"Grid#"	"# DHRUs inside this Grid#"
5		"Row-ID"	
3		"Col-ID"	
1.0		"A-grid"	
3	4	"Grid#"	"# DHRUs inside this Grid#"

2	2	3	3	“Row-ID”	“Row-ID”	“Row-ID”	“Row-ID”
3	4	3	4	“Col-ID”	“Col-ID”	“Col-ID”	“Col-ID”
0.125	0.125	0.375	0.375	“A-grid”	“A-grid”	“A-grid”	“A-grid”

EXPLANATION OF EXAMPLE FILE CONTENTS

- There are a total of 400 DHRUs in the coupled model and the DHRU with the most grid cells contributing to it has 7.
- The first DHRU has 0 grid cells which contribute to it. This is likely because the DHRU derived from the SWAT model is located outside of the boundaries of the MODFLOW model.
    - When a DHRU has no grid cells contributing to it, leave 3 blank lines before the next entry.
- The second DHRU has 2 grid cells which contribute to it.
    - The first grid cell contributing to DHRU-2 is row number 5
    - The first grid cell contributing to DHRU-2 is column number 3
    - The first grid cell comprises 100% of the area of DHRU-2
- The third DHRU has 4 grid cells which contribute to it.
    - The first grid cells contributing to DHRU-3 is row number 2
    - The second grid cells contributing to DHRU-3 is row number 2
    - The third grid cells contributing to DHRU-3 is row number 3
    - The fourth grid cells contributing to DHRU-3 is row number 3
    - The first grid cells contributing to DHRU-3 is column number 3
    - The second grid cells contributing to DHRU-3 is column number 4

  - The third grid cells contributing to DHRU-3 is column number 3
  - The fourth grid cells contributing to DHRU-3 is column number 4
  - The first grid cells comprises 12.5% of the area of DHRU-3
  - The second grid cells comprises 12.5% of the area of DHRU-3
  - The third grid cells comprises 37.5% of the area of DHRU-3
  - The fourth grid cells comprises 37.5% of the area of DHRU-3
  - Because this percent area row does not add up to 100% this DHRU is likely only partially covered by the MODFLOW model, which is allowed with this coupling.

EXPLANATION OF: "map_river2grid.txt"

This file contains an entry for every MODFLOW river cell as provided by MODFLOW's river package file. The file starts with the first river cell and ends with last river cell (i.e. 1 to number of river grid cells). Listed below are terms used in the next section to label the content of the file.

- DHRU = Disaggregated Hydrologic Response Unit
- ID = identification number
- T-RCells = total number of river cells in MODFLOW
- R-Grid# = the river grid cell #
- N-Grid = the # unique rivers (from different SWAT sub-basins) which are contained by this grid cell
- RivID = river IDs, which contribute to the current river cell
- L-Riv = length of river within current "R-Grid#"

EXAMPLE FILE CONTENTS, WITH LABELS:

4		“T-RCells”	
1	1	“R-Grid#”	“N-Grid”
9		“RivID”	
70.7107		“L-Riv”	
2	2	“R-Grid#”	“N-Grid”
9	8	“RivID”	“RivID”
100.0	212.132	“L-Riv”	“L-Riv”

EXPLANATION OF EXAMPLE FILE CONTENTS

- There are a total of 4 river cells in the MODFLOW model
- The first MODFLOW river cell has 1 SWAT river contained within it.
    - The SWAT river number, which is also the sub-basin ID, for this river is “9”
    - The length of the SWAT river within this grid cell is “70.71070” meters
- The second MODFLOW river cell has 1 SWAT rivers contained within it.
    - The first SWAT river number that contributes to this MODFLOW river cell is “9”
    - The second SWAT river number that contributes to this MODFLOW river cell is “8”
    - The length of the first SWAT river within this grid cell is “100.0” meters
    - The length of the second SWAT river within this grid cell is “212.132” meters

APPENDIX IV: SWAT-MODFLOW FORTAN CODE

MODEVENT.F

```
      module modevent
!!    ~ ~ ~ Author ~ ~ ~
!!    Andre Dozier, PhD student
!!    Colorado State University 2012-2016
!!    Comment initials "aqd"
!!
!!    ~ ~ ~ PURPOSE ~ ~ ~
!!    Implements a generic code event that can be subscribed to by any number of
subscribers with two different structures:
!!     - a subroutine with the following format can be subscribed to an event by "call
my_event%subscribe(my_subscriber)"
!!         subroutine my_subscriber(eventdata)
!!             class (ieventdata), pointer :: eventdata
!!             <my_work>
!!         end subroutine
!!
!!     - a subclass with the following format can be subscribed to an event by "call
my_event%subscribe_object(my_subscriber)"
!!         type, public, extends(isubscriber) :: my_subscriber_class
!!             <my_data>
!!         contains
!!             procedure, pass :: run => my_subscriber_subroutine
!!         end type
!!
!!         subroutine my_subscriber_subroutine(subsc, eventdata)
!!             class (my_subscriber_class), intent(inout) :: subsc
!!             class (ieventdata), pointer :: eventdata
!!             <my_work>
!!         end subroutine
!!
!!    An event can be instantiated by "type(event), target :: my_event"
!!    An event can be "fired" by "call my_event%fire()".
!!
      implicit none

        ! Parameters
        logical :: verbose = .false.

        ! Define a generic subscriber
        type, public, abstract :: isubscriber
```

```
    integer :: ind = -1
    class (isubscriber), pointer :: next => null()
contains
    procedure(abstrun), deferred, pass :: run
end type

! Define the interface for event data
type, public, abstract :: ieventdata
end type

! Define abstrun
abstract interface
    subroutine abstrun(subsc, eventdata)
        import :: isubscriber
        import :: ieventdata
        class (isubscriber), intent(inout) :: subsc
        class (ieventdata), pointer :: eventdata
    end subroutine
end interface

! Running a simplified subscriber's subroutine
interface
    subroutine simple(eventdata)
        import :: ieventdata
        class (ieventdata), pointer :: eventdata
    end subroutine simple
end interface

! Define the simplified subscriber
type, public, extends(isubscriber) :: subscriber
    procedure (simple), pointer, nopass :: simple => null()
contains
    procedure, pass :: run => simplerun
end type subscriber

! Define the event class
type, public :: event
    integer :: subindex = 0
    class (ieventdata), pointer :: data => null()
    class (isubscriber), pointer :: first => null()
    class (isubscriber), pointer :: last => null()
contains
    procedure :: subscribe
    procedure :: subscribe_object
    procedure :: unsubscribe
    procedure :: fire
```

```
  end type event

contains

  ! Run the subscriber's event
  subroutine simplerun(subsc, eventdata)
    class (subscriber), intent(inout) :: subsc
    class (ieventdata), pointer :: eventdata

    if (associated(subsc%simple)) call subsc%simple(eventdata)
  end subroutine

  ! Subscribe to the event
  function subscribe(self, newsubscriber)
    integer :: subscribe
    class(event), intent(inout) :: self
    type (subscriber), pointer :: subsc
    class(isubscriber), pointer :: isubsc

    interface
      subroutine newsubscriber(eventdata)
        import :: ieventdata
        class (ieventdata), pointer :: eventdata
      end subroutine newsubscriber
    end interface

    ! set the subscriber's run method to the new method
    allocate(subsc)
    subsc%simple => newsubscriber

    ! allocate the subscriber pointer
    allocate(isubsc, source=subsc)

    ! subscribe the object
    call self%subscribe_object(isubsc)

    ! return the index of the subscriber (for unsubscription later)
    subscribe = isubsc%ind
  end function

  ! Subscribes a subscriber object
  subroutine subscribe_object(self, newsubscriber)
    class(event), intent(inout) :: self
    class(isubscriber), pointer :: newsubscriber

    ! assign the subscribers index
    newsubscriber%ind = self%subindex
    if (verbose) print *,'adding index ',self%subindex

    ! ensure the first subscriber is allocated
    if (.not.associated(self%first)) then
        self%first => newsubscriber
        self%last => newsubscriber
    else
        self%last%next => newsubscriber
        self%last => self%last%next
    end if

    ! increment the index of the subscriber
    self%subindex = self%subindex + 1
end subroutine

! Unsubscribe to the event
subroutine unsubscribe(self, unsubi)
    class(event), intent(inout) :: self
    integer :: unsubi
    class (isubscriber), pointer :: prev, sub

    ! exit if there are no subscribers
    if (.not.associated(self%first)) return

    ! find the subscriber with the same index
    prev => null()
    sub => self%first
    if (verbose) print *,'looking for index ',unsubi
    do while (sub%ind /= unsubi)
        if (.not.associated(sub%next)) return
        prev => sub
        sub => sub%next
    end do

    ! remove the subscriber and deallocate it
    if (associated(prev)) then
        ! remove sub
        if (associated(sub%next)) then
            ! point to the next subscriber
            prev%next => sub%next
        else
            ! point to the second to last subscriber
            if (verbose) print *,'removing end subscriber'
            deallocate(prev%next)
            self%last => prev
```

```
      end if
      deallocate(prev)
      deallocate(sub)
    else
      ! delete the first subscriber
      if (associated(sub%next)) then
        ! point to the second subscriber if existent
        if (verbose) print *,'removing first subscriber'
        self%first => sub%next
        deallocate(sub)
      else
        ! the first subscriber is the only one left
        if (verbose) print *,'removing all subscribers'
        deallocate(self%first)
      end if
    end if
  end subroutine

  ! Fire the event
  subroutine fire(self)
    class(event), intent(in) :: self
    integer :: i
    class (isubscriber), pointer :: sub, nextsub

    ! exit if there are no subscribers
    if (.not.associated(self%first)) return

    ! run all the subscribers
    sub => self%first
    do  ! while (associated(sub%next))
      if (associated(sub%next)) nextsub => sub%next
      call sub%run(self%data)
      if (.not.associated(sub%next)) return
      sub => nextsub
    end do
    deallocate(sub)
  end subroutine

end module modevent
```

PKG_MODFLOW.F

```
      subroutine pkg_modflow
!!    ~ ~ ~ Author ~ ~ ~
!!    Tyler Wible, Masters student
!!    Colorado State University 2012-2014
!!   Comment initials "tcw"
!!
!!   Andre Dozier, PhD student
!!   Colorado State University 2012-2016
!!   Comment initials "aqd"
!!
!!   ~ ~ ~ PURPOSE ~ ~ ~
!!   Adds the SM package to MODFLOW
!!
      use modevent !event class
      use global, only: OnRivPkg,OnRechPkg,OnUZFPkg,MFrunTop !MODFLOW
events
      implicit none

      integer :: pkg_i

      interface
         subroutine sm_mfRiver(eventdata)
            import :: ieventdata
            class(ieventdata), pointer :: eventdata
         end subroutine
      end interface

      interface
         subroutine sm_recharge(eventdata)
            import :: ieventdata
            class(ieventdata), pointer :: eventdata
         end subroutine
      end interface

      interface
         subroutine sm_uzf(eventdata)
            import :: ieventdata
            class(ieventdata), pointer :: eventdata
         end subroutine
      end interface

      interface
         subroutine sm_mf_read(eventdata)
            import :: ieventdata
            class(ieventdata), pointer :: eventdata
         end subroutine
      end interface

      ! River stage is provided by SWAT
      pkg_i = OnRivPkg%subscribe(sm_mfRiver)
```

```
        ! Recharge is provided by SWAT
        pkg_i = OnRechPkg%subscribe(sm_recharge)

        ! Infiltration and ET is provided by SWAT
        pkg_i = OnUZFPkg%subscribe(sm_uzf)

        ! If SM is used, de-activate mf_read and SLMT7PNT insided of mf_run because it
    has already been called by the linkage
        pkg_i = MFrunTop%subscribe(sm_mf_read)

      end subroutine pkg_modflow
```

PKG_SWAT.F

```
      subroutine pkg_swat
!!    ~ ~ ~ Author ~ ~ ~
!!    Tyler Wible, Masters student
!!    Colorado State University 2012-2014
!!    Comment initials "tcw"
!!
!!    Andre Dozier, PhD student
!!    Colorado State University 2012-2016
!!    Comment initials "aqd"
!!
!!    Ryan Bailey, Post-Doc student (2012-2013), Assistant Professor (2013-)
!!    Colorado State University 2012-
!!    Comment initials "rtb"
!!
!!    ~ ~ ~ PURPOSE ~ ~ ~
!!    Adds the SM package to SWAT replacing the groundwater modeling
!!    component with MODFLOW
!!
        use modevent !event class
        use parm, only: OnInit,normOutUpdate,OnCommand19,OnSoilTempCalc,
     &                 OnCropGrowthCalc,OnRouteInitialize !SWAT events
        use sm_getgw !Groundwater Related events
        implicit none

        integer :: pkg_i

        interface
          subroutine sm_init_mf(eventdata)
            import :: ieventdata
            class (ieventdata), pointer :: eventdata
          end subroutine
        end interface

        interface
           subroutine sm_mf_run(eventdata)
              import :: ieventdata
              class (ieventdata), pointer :: eventdata
           end subroutine
        end interface

        interface
           subroutine sm_normOut(eventdata)
              import :: ieventdata
              class (ieventdata), pointer :: eventdata
           end subroutine
        end interface

        ! Initialize modflow
        pkg_i = OnInit%subscribe(sm_init_mf)

        !Check if SWAT should update it's groundwater output variables
        pkg_i = normOutUpdate%subscribe(sm_normOut)

        ! Call MODFLOW
        pkg_i = OnCommand19%subscribe(sm_mf_run)

        !Calculate GW contribution from MODFLOW
        pkg_i = OnSoilTempCalc%subscribe(sm_getgwcontr)

        !Compute nutrient loading from MODFLOW
        pkg_i = OnCropGrowthCalc%subscribe(sm_getgwnutr)

        !Determine if initializing the swat reach loss variable is necessary
        pkg_i = OnRouteInitialize%subscribe(sm_getrechloss)

      end subroutine pkg_swat
```

SM_ALLOCATE.F

```
      subroutine sm_allocate()
!!    ~ ~ ~ Author ~ ~ ~
!!    Tyler Wible, Masters student
!!    Colorado State University 2012-2014
!!    Comment initials "tcw"
!!
!!    ~ ~ ~ PURPOSE ~ ~ ~
!!    this subroutine allocates array sizes for variables used in the SM linkage between
SWAT, MODFLOW

      use parm, only: msub !SWAT
      use sm_parm !sm linkage
      implicit none

!     Allocate the size of the river-to-grid variables
      allocate(grid2riv_id(nriver_cells, msub))
      allocate(grid2riv_len(nriver_cells, msub))
      grid2riv_id = 0.
      grid2riv_len = 0.

!     Allocate the size of disaggregated hru variables before MODFLOW and UZF-RT3D
runs
      allocate(etremain_dhru(dhru))
      allocate(sepbtm_dhru(dhru))
      etremain_dhru = 0.
      sepbtm_dhru = 0.

      return
      end
```

SM_CLOSE.F

```
      subroutine sm_close
!!    ~ ~ ~ Author ~ ~ ~
!!    Tyler Wible, Masters student
!!    Colorado State University 2012-2014
!!    Comment initials "tcw"
!!
!!    ~ ~ ~ Purpose ~ ~ ~
!!    This subroutine deallocates the variables from sm_parm and some
!!    additional MODFLOW which had their deallocate calls moved

!     Import variables
      use GWFRIVMODULE, only: RIVAUX, RIVR !MODFLOW
      use sm_parm !sm linkage
      implicit none

!     Deallocate MODFLOW variables
      deallocate(RIVAUX)
      deallocate(RIVR)

!     Deallocate sm variables
      deallocate(g2d_r)
      deallocate(g2d_c)
      deallocate(g2d_area)
      deallocate(d2g_id)
      deallocate(d2g_area)
      deallocate(d2h_id)
      deallocate(d2h_area)
      deallocate(grid2riv_id)
      deallocate(grid2riv_len)

      deallocate(etremain_dhru)
      deallocate(sepbtm_dhru)

      return
      end
```

SM_CONVERSION2MF.F

```
      subroutine sm_conversion2mf
!!    ~ ~ ~ Author ~ ~ ~
!!    Tyler Wible, Masters student
!!    Colorado State University 2012-2014
!!    Comment initials "tcw"
!!
!!    ~ ~ ~ PURPOSE ~ ~ ~
!!    This subroutine converts the necessary SWAT variables (per hru) into SM variables
!!    (per disaggregated hru, dhru) then converts these into the proper units for
MODFLOW
!!
!!    ~ ~ ~ Variables Used~ ~ ~
!!    name            |units                |definition
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!    LENUNI          |unit_in              |the MODFLOW variable for which length units are
being used
!!    ITMUNI          |unit_out             |the MODFLOW variable for which time units are
being used
!!    sepbtm(:)       |mm H2O               |percolation from bottom of soil profile for
!!                    |                     |the day in HRU
!!    etremain(:)     |mm H2O               |remaining et to be passed from SWAT to
!!                    |                     |MODFLOW-UZF, etremain(j) = pet_day-etday
!!    etremain_dis(:) |mm H2O          (in) |remaining et to be passed from SWAT to
!!                    |LENUNI**3/ITMUNI (out)|MODFLOW-UZF per disaggregated hru
(dhru)
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!
!!    ~ ~ ~ Variables Modified ~ ~ ~
!!    name            |units                |definition
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!    sepbtm_dhru(:)  |mm H2O          (in) |percolation from bottom of the soil profile
!!                    |                     |populated from sepbtm and sm_hru2dhru conversion
!!                    |LENUNI**3/ITMUNI (out)|for the day in HRU, now in MODFLOW units
!!    etremain_dis(:) |mm H2O          (in) |remaining et to be passed from SWAT to
!!                    |LENUNI**3/ITMUNI (out)|MODFLOW-UZF per disaggregated hru
(dhru)
!!                    |                     |populated from etremain and sm_hru2dhru conversion
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!
!!    ~ ~ ~ Local Definitions ~ ~ ~
!!    name          |units                |definition
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!    mf_lengthUnit |MODFLOW length unit   |the modified inteteger to represent
!!                  |                      |the MODFLOW unit of length for units
!!    mf_timeUnit   |MODFLOW time unit     |the modified integer to represent
!!                  |                      |the MODFLOW unit of time for units
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!
!!    ~ ~ ~ ~ ~ ~ END SPECIFICATIONS ~ ~ ~ ~ ~ ~

      !Import variables
      use parm, only:sepbtm,etremain,leapyr  !SWAT
      use GLOBAL, only:LENUNI,ITMUNI !MODFLOW
      use sm_parm !sm linkage
      implicit none

      !Define local variables
      integer mf_lengthUnit,mf_timeUnit
      mf_lengthUnit = LENUNI + 10
      mf_timeUnit = ITMUNI

      !Convert SWAT hru based variables into SM dis-aggregated hrus for more accurate
surface/groundwater interaction
      call sm_hru2dhru(sepbtm, sepbtm_dhru)
      call sm_hru2dhru(etremain, etremain_dhru)

      !Convert SM variables into MODFLOW units
      call units(sepbtm_dhru, 14, mf_lengthUnit, 1, dhru, leapyr)! to convert units (mm/day
to LENUNI/day)
      call units(sepbtm_dhru, mf_timeUnit, 4, 1, dhru, leapyr)! to convert time units
(LENUNI/days to LENUNI/ITMUNI)
      call units(etremain_dhru, 14, mf_lengthUnit, 1, dhru, leapyr)! to convert length units
(mm/day to LENUNI/day)
      call units(etremain_dhru, mf_timeUnit, 4, 1, dhru, leapyr)! to convert time units
(LENUNI/days to LENUNI/ITMUNI)
```

```
      return
      end
```

SM_CONVERSION2SWAT.F

```
      subroutine sm_conversion2swat
!!    ~ ~ ~ Author ~ ~ ~
!!    Tyler Wible, Masters student
!!    Colorado State University 2012-2014
!!    Comment initials "tcw"
!!
!!    ~ ~ ~ Purpose ~ ~ ~
!!    This subroutine converts all the MODFLOW variables, to pass into SWAT, into the
proper units and SWAT variables
!!
!!    ~ ~ ~ Variables Used ~ ~ ~
!!    name          |units          |definition
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~
!!    gw_q(:)       |mm H2O         |groundwater contribution to streamflow from
!!                  |               |HRU on current day
!!    gw_qdeep(:)   |mm H2O         |groundwater contribution to streamflow from deep
aquifer
!!                  |               |from HRU on current day
!!    gwht(:)       |m              |groundwater height
!!                  |               |HRU on current day
!!    hru_fr(:)     |none           |fraction of subbasin area in HRU
!!    hrutot(:)     |none           |number of HRUs in subbasin
!!    leapyr        |none           |leap year flag
!!                  |               |0  leap year
!!                  |               |1  regular year
!!    msub          |none           |max number of subbasins
!!    nhru          |none           |number of HRUs in watershed
!!    rttlc(:)      |m^3 H2O        |transmission losses from reach on day
!!    sub_fr(:)     |none           |fraction of watershed area in subbasin
!!    sub_km(:)     |km^2           |area of subbasin in square kilometers
!!    LENUNI        |MODFLOW length |the MODFLOW variable for which length
units are being used
!!    ITMUNI        |MODFLOW time   |the MODFLOW variable for which time units
are being used
!!    NROW          |n/a            |the MODFLOW variable for number of rows of grids
!!    NCOL          |n/a            |the MODFLOW variable for number of columns of grids
!!    NLAY          |n/a            |the MODFLOW variable for number of layers of grids
!!    HNEW(:,:,:)   |LENUNI         |MODFLOW variable for the new head in the aquifer
!!    BOTM(:,:,:)   |LENUNI         |MODFLOW variable for the elevation of the
!!                |               |bottom of each grid cell layer
!!   RIV(6,:)       |LENUNI         |MODFLOW variable for properties of the river cells
!!                |n/a            |(1,:) cell layer index
!!                |n/a            |(2,:) cell row index
!!                |n/a            |(3,:) cell column index
!!                |LENUNI         |(4,:) cell river stage (bottom elevation + depth)
!!                |LENUNI/ITMUNI    |(5,:) cell conductance
!!                |LENUNI         |(6,:) cell river bottom elevation
!!   grid2riv_id    |none           |a list per MODFLOW river grids (cols) containing
!!                |               |the SWAT river IDs within that grid cell
!!   grid2riv_len    |m              |a list per MODFLOW river grids (cols) containing
!!                |               |the SWAT river lengths within that grid cell
!!   nriver_cells    |n/a            |the total number of river grid cells in MODFLOW
!!   ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~
!!
!!   ~ ~ ~ Variables Modified ~ ~ ~
!!   name           |units               |definition
!!   ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~
!!   gwht(:)        |m                   |groundwater height SWAT variable, populated by
MODFLOW's HNEW
!!   gw_q(:)        |mm H2O              |groundwater contribution to streamflow from
!!                |                    |HRU on current day
!!   ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~
!!
!!   ~ ~ ~ Local Definitions ~ ~ ~
!!   name           |units               |definition
!!   ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~
!!   mf_lengthUnit   |MODFLOW length unit   |the modified integer to represent
!!                |                    |the MODFLOW unit of length for units
!!   mf_timeUnit     |MODFLOW time unit     |the modified integer to represent
!!                |                    |the MODFLOW unit of time for units
!!   IL             |none                |the layer index for getting info. from RIVR
!!   IC             |none                |the column index for getting info. from RIVR
!!   IR             |none                |the row index for getting info. from RIVR
!!   h              |none                |the hru index for looping over subbasin's hrus
!!   sum_rivrate(1)  |LENUNI**3/ITMUNI      |holder for the sum of river rates
contributing to a
!!                |                    |given subbasin to be mapped to rivSegPar
!!   wtlocation(:,:) |n/a                 |sm variable of the rows and columns of
!!                |                    |MODFLOW grid cells with a value indicating
!!                |                    |which layer the water table is located in
!!    gw_tot          |m**3/day             |The total amount of groundwater discharge from the
current
!!                    |                     |grid cell to the subbasin outlet
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~
!!    ~ ~ ~ ~ ~ ~ End Specifications ~ ~ ~ ~ ~ ~

      !Import variables
      use parm, only: gw_q,gw_qdeep,gwht,hru_fr,hru1,hrutot, !SWAT
     &               leapyr,msub,nhru,rttlc,sub_fr,sub_km
      use GLOBAL,only:LENUNI,ITMUNI,NROW,NCOL,NLAY,HNEW,BOTM
!MODFLOW
      use GWFRIVMODULE, only:RIVR !MODFLOW
      use sm_parm !sm linkage
      implicit none

      !define local variables
      integer mf_lengthUnit,mf_timeUnit,subID,dhruID
      integer IL,IC,IR,i,j,k,h
      DOUBLE PRECISION HHNEW,CHRIV,RRBOT,CCRIV
      real grid_rivlen, HRIV, CRIV, RBOT, RATE, gw_tot
      real wtlocation(NCOL,NROW)
      real rt_cnew_location(NCOL,NROW)
      real sum_rivrate(1)
      real gwht_dhru(dhru)
      mf_lengthUnit = LENUNI + 10
      mf_timeUnit = ITMUNI
      wtlocation = 1 !Set default water table location
      subID = 0
      gw_tot = 0
      gwht_dhru = 0
      h = 1 !set starting hru index

      !Loop through MODFLOW variables and pull out information
      do j=1, NROW
        do i=1, NCOL
          do k=1, NLAY
            !Loop through and find which layer the water table is in
            if(HNEW(i,j,k).LT.BOTM(i,j,k-1) .and. !BOTM(J,I,0) contains the top of layer1
     &          HNEW(i,j,k).GT.BOTM(i,j,k)) then
              wtlocation(i,j) = k
            endif
          enddo
        enddo
      enddo
      !Preprocess replacing the values of gw_q and gw_qdeep which will be populate with
data from MODFLOW
      rttlc = 0.
      do i=1, nhru
        do j=1, d2h_size
          dhruID = d2h_id(i,j)
          if(dhruID.ne.0)then
            gw_q(i) = 0
            gw_qdeep(i) = 0 !also zero the deep groundwater flow to streams because
MODFLOW only passes back 1 baseflow value
          endif
        enddo
      enddo

      do i=1, nriver_cells
        !reset averaged river properties
        grid_rivlen = 0.
        sum_rivrate(1) = 0.
        RATE = 0.

        !Use original MODFLOW code to calculate the rate of loss/gain from/to rivers
(LENUNI^3/ITMUNI)
        !the below code is borrowed from the MODFLOW subroutine LMT7RIV7 to
calculate RATE
        IL=RIVR(1,i)
        IR=RIVR(2,i)
        IC=RIVR(3,i)
        !--GET RIVER PARAMETERS FROM RIVER LIST.
        HRIV=RIVR(4,i)
        CRIV=RIVR(5,i)
        RBOT=RIVR(6,i)
        HHNEW=HNEW(IC,IR,IL)
        CHRIV=CRIV*HRIV
        CCRIV=CRIV
        RRBOT=RBOT
        !--COMPARE HEAD IN AQUIFER TO BOTTOM OF RIVERBED.
        IF(HHNEW.GT.RRBOT) RATE=CHRIV-CCRIV*HHNEW !--AQUIFER HEAD
> BOTTOM THEN RATE=CRIV*(HRIV-HNEW).
        IF(HHNEW.LE.RRBOT) RATE=CRIV*(HRIV-RBOT) !--AQUIFER HEAD <
BOTTOM THEN RATE=CRIV*(HRIV-RBOT)
        !end MODFLOW code

        !Get total segments length within current grid cell
        do j=1, msub
          grid_rivlen = grid_rivlen + grid2riv_len(i,j)
        enddo
```

```
        !Apply only a portion of the river seepage/discharge rate to each river segment
within the
        !current grid cell based on the relative length of the river segment within the grid
cell
        do j=1, msub !j should only be used as an index for grid2riv_id and grid2riv_len, but
NOT as the actual subbasin index
          !determine if the current river cell is part of this subbasin
          subID = grid2riv_id(i,j)

          if(subID.ne.0)then
            sum_rivrate(1) = RATE * (grid2riv_len(i,j)/grid_rivlen)
            !for example, if subID's river is all of the river length in the grid, the
            !subbasin would get 100% of the river rate, if 2 rivers (from 2 subbasins) of
            !equal length are within the grid, each subbasin would get 50% of the river rate

            if(sum_rivrate(1).lt.0) then !a negative rate indicates water leaving the aquifer,
aka discharge to river
              !take the groundwater discharge rate and convert it into SWAT units
              sum_rivrate(1) = -sum_rivrate(1)
              call units(sum_rivrate, mf_lengthUnit, 15, 3, 1, leapyr)! convert length unit
(LENUNI**3/ITMUNI to km**3/ITMUNI)
              call units(sum_rivrate, 4, mf_timeUnit, 1, 1, leapyr)! convert time unit
(km**3/ITMUNI to km**3/day)
              call units(sum_rivrate, 15, 14, 1, 1, leapyr)!convert length unit (km**3/day to
mm-km**2/day)
              gw_tot = sum_rivrate(1)/sub_km(subID)!divide by the subbasin area, in km,
thus finishes converting units (mm-km**2/day to mm/day)

              !determine the starting HRU index for subID's HRUs
              h = hru1(subID)
              do k=1, hrutot(subID)
                !apply a portion of gw_tot to gw_q based on the % hru area in subID's subbasin
area
                gw_q(h)= gw_q(h) + (hru_fr(h)*sub_fr(subID))*gw_tot
                h = h + 1
              enddo

            else !a positive rate indicates water entering the aquifer, aka leakage from river
              !take the seepage to groundwater rate and convert it into SWAT units
              call units(sum_rivrate, mf_lengthUnit, 12, 3, 1, leapyr)! convert length unit
(LENUNI**3/ITMUNI to m**3/ITMUNI)
              call units(sum_rivrate, 4, mf_timeUnit, 1, 1, leapyr)! convert time unit
(m**3/ITMUNI to m**3/day)

              rttlc(subID) = rttlc(subID) + sum_rivrate(1)
          endif
        endif

      enddo
    enddo

    !Convert MODFLOW grid variable into SM dis-aggregated hru variable
    call sm_grid2dhru3D(HNEW, gwht_dhru,
wtlocation)!!!!!!!!!!!!!!!!!!!!!!!!!!!!!reconsider this since SWAT never uses the value of
variable!!!!!!!!!!!!!!!!tcw

    !Convert SM variable into SWAT units
    call units(gwht_dhru, mf_lengthUnit, 12, 1, dhru, leapyr)!convert length unit
(LENUNI to m)

    !Convert SM variable into SWAT variable
    call sm_dhru2hru(gwht_dhru, gwht, 0)

    !Call additional MODFLOW to SWAT subfunctions
    call sm_upflux_to_soil

    !Update SWAT's output variables for output.std with the new groundwater results
from MODFLOW
    call sm_updateOutput

    return
    end
```

SM_DHRU2GRID2D.F

```
    subroutine sm_dhru2grid2D (smVar, mfVar2, mult_TF)
!!    ~ ~ ~ Author ~ ~ ~
!!    Tyler Wible, Masters student
!!    Colorado State University 2012-2014
!!    Comment initials "tcw"
!!
!!    ~ ~ ~ Purpose ~ ~ ~
!!    This subroutine converts SM-based disaggregated HRUs (dhrus) to 2D MODFLOW-
based grids.
!!    Additionally, this subroutine multiplies a MODFLOW 2D variable by the grid area
of each
!!    MODFLOW grid if mult_TF is 1, divides a MODFLOW 2D variable by the grid area
of each
!!    MODFLOW grid if mult_TF is 2, or does nothing additional if mult_TF is 0.
!!
!!    ~ ~ ~ Variables Used ~ ~ ~
!!   name        |units        |definition
!!   ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!   d2g_id      |none         |Array per MODFLOW grid of the IDs of the
!!               |             |dhrus which contribute to this grid
!!   d2g_area    |none         |Array per MODFLOW grid of the percent area of
!!               |             |the dhrus which contribute to this grid
!!   d2g_size    |none         |the maximum number of dhrus which contribute
!!               |             |to a single grid, used for looping and
!!               |             |dimensioning variables
!!   smVar       |unknown      |sm variable (list per dhru)
!!   mfVar2      |unknown      |2D MODFLOW variable (NCOL, NROW) to be
!!               |             |populated with the contents of the SM variable
!!   mult_TF     |n/a          |integer, if "1" will multiply array
!!               |             |by the grid area, if "2" will divide
!!               |             |array by the grid area, if "0" will not
!!               |             |convert the variable based on MODFLOW area
!!   DELR        |LENUNI       |MODFLOW variable, the thickness of the row per column
!!   DELC        |LENUNI       |MODFLOW variable, the thickness of the column per row
!!   ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!
!!   ~ ~ ~ Variables Modified ~ ~ ~
!!   name        |units        |definition
!!   ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!   mfVar2      |unknown      |2D MODFLOW variable (NCOL, NROW) now populated
!!               |             |with the contents of the SM variable
!!   ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!
!!   ~ ~ ~ Local Definitions ~ ~ ~
!!   name        |units        |definition
!!   ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!   newVar2     |unknown      |2D MODFLOW variable (NCOL, NROW) that is
!!               |             |temporarily populated with the contents of
!!               |             |the SM variable before checking for remaining
!!               |             |grid area not contributed to by a dhru
!!   areaUsed    |%            |2D MODFLOW variable (NCOL, NROW) that is
!!               |             |used to track how much of the grid area has
!!               |             |been contributed to by the dhrus so that any
!!               |             |remaining grid area not contributed to by a dhru
!!               |             |uses the original MODFLOW value
!!   i           |n/a          |MODFLOW row loop index
!!   j           |n/a          |MODFLOW column loop index
!!   ctr         |n/a          |MODFLOW grid id index
!!   k           |n/a          |d2g_size loop index
!!   cellUsed    |n/a          |a true/false for whether the MODFLOW grid
!!               |             |has interacted with a SM dhru and should
!!               |             |therefore be converted for area/unit reasons
!!   ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!   ~ ~ ~ ~ ~ ~ END SPECIFICATIONS ~ ~ ~ ~ ~ ~

!    Import variables
     use GLOBAL, only: NCOL,NROW,DELR,DELC !MODFLOW
     use sm_parm, only: dhru,d2g_size,d2g_id,d2g_area !sm linkage
     implicit none

!    Define local variables
     real smVar(dhru)
     real mfVar2(NCOL,NROW)
     real newVar2(NCOL,NROW)
     real areaRemain(NCOL,NROW)
     integer mult_TF, i, j, k, ctr, dhruID
     logical cellUsed

!    Initialize local variables
     newVar2 = 0.
     areaRemain = 1.
     ctr = 1
     cellUsed = .false.

!    Convert SM dhru variables into MODFLOW-grid variables by multiplying each
!    contributing dhru variable by its percent area contributing to each grid
     do j=1, NROW
       do i=1, NCOL
         do k=1, d2g_size
           dhruID = d2g_id(ctr,k)
           if(dhruID.ne.0)then
             !Convert the dhru information into grid information
             newVar2(i,j) =newVar2(i,j)+smVar(dhruID)*d2g_area(ctr,k)
             !Track how much of the area of the grid has been contributed to (%)
             areaRemain(i,j) = areaRemain(i,j) - d2g_area(ctr,k)
             cellUsed = .true.
           endif
         enddo

         !If the grid cell was contributed to by dhrus, and
         !the units need to convert the area, do so here
         if(cellUsed .and. mult_TF.eq.1)then
           !Multiply variable by cell area
           newVar2(i,j) = newVar2(i,j)*DELR(i)*DELC(j)
         else if(cellUsed .and. mult_TF.eq.2)then
           !Divide variable by cell area
           newVar2(i,j) = newVar2(i,j)/(DELR(i)*DELC(j))
         endif
```

```
          !Store the converted dhru results into the MODFLOW variable
          if(areaRemain(i,j) > 0)then
            !If the dhrus do not completely cover the grid cell, retain a portion of the original
grid data
            mfVar2(i,j) = newVar2(i,j) + mfVar2(i,j)*areaRemain(i,j)
          else
            mfVar2(i,j) = newVar2(i,j)
          endif

          !reset the counters for the loop
          cellUsed = .false.
          ctr = ctr + 1
        enddo
      enddo

      return
      end
```

SM_DHRU2HRU.F

```
      subroutine sm_dhru2hru (smVar, swatVar, mult_TF)
!!    ~ ~ ~ Author ~ ~ ~
!!    Tyler Wible, Masters student
!!    Colorado State University 2012-2014
!!    Comment initials "tcw"
!!
!!    ~ ~ ~ Purpose ~ ~ ~
!!    This subroutine converts SM-based disaggregated HRUs (dhru) to SWAT HRUs.
!!    Additionally, this subroutine multiplies a SWAT variable by the area, in
!!    km**2, of each SWAT hru if mult_TF is true 1, divides a SWAT variable by
!!    the area, in km**2, of each SWAT hru if mult_TF is false 2, or does nothing
!!    additional if mult_TF is 0.
!!
!!    ~ ~ ~ Variables Used ~ ~ ~
!!    name        |units         |definition
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!    d2h_id      |none          |Array per SWAT hru of the IDs of the dhrus
!!                |              |which contribute to this hru
!!    d2h_area    |none          |Array per SWAT hru of the percent area of
!!                |              |the dhrus which contribute to this hru
!!    d2h_size    |none          |the maximum number of dhrus which contribute
!!                |              |to a single hru, used for looping and
!!                |              |dimensioning variables
!!    smVar       |unknown       |SM variable (list of dhrus)
!!    swatVar     |unknown       |SWAT variable (list of nhru) to be populated
!!             |             |with the contents of the SM variable
!!   hru_km(:)  |km**2        |area of HRU in square kilometers
!!   ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!
!!   ~ ~ ~ Variables Modified ~ ~ ~
!!   name        |units        |definition
!!   ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!   swatVar     |unknown      |SWAT variable (list of nhru) to be populated
!!             |             |with the contents of the SM variable
!!   ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!
!!   ~ ~ ~ Local Definitions ~ ~ ~
!!   name        |units        |definition
!!   ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!   i           |none         |SWAT HRU loop index
!!   j           |none         |SM dhru loop index
!!   cellUsed    |n/a          |a true/false for whether the SWAT hru
!!             |             |has interacted with a SM dhru and should
!!             |             |therefore be converted for area/unit reasons
!!   ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

!!   ~ ~ ~ ~ ~ ~ END SPECIFICATIONS ~ ~ ~ ~ ~ ~

!    Import variables
     use parm, only: nhru, hru_km !SWAT
     use sm_parm, only: dhru, d2h_size, d2h_id, d2h_area !sm linkage
     implicit none

!    Define local variables
     real smVar(dhru)
     real swatVar(nhru)
     integer mult_TF, i, j, dhruID
     logical cellUsed

!    Initialize local variables
     cellUsed = .false.

!    Convert SM-dhru variables into SWAT-HRU variables by multiplying each
!    contributing dhru variable by its percent area contributing to each HRU
     do i=1, nhru
       do j=1, d2h_size
         dhruID = d2h_id(i,j)
         if(dhruID.ne.0)then

           if(cellUsed)then
             !if this is the second or more time referencing this cell, add to its contents
```

```
            swatVar(i) = swatVar(i) + smVar(dhruID)*d2h_area(i,j)
          else
            !if this is the first time referencing this cell, overwrite its contents
            swatVar(i) = smVar(dhruID)*d2h_area(i,j)
            cellUsed = .true.
          endif

        endif
      enddo

      !If the hru was contributed to by dhrus, and
      !the units need to convert the area, do so here
      if(cellUsed .and. mult_TF.eq.1)then
        !Multiply variable by cell area, in km
        swatVar(i)=swatVar(i)*hru_km(i)
      else if(cellUsed .and. mult_TF.eq.2)then
        !Divide variable by cell area, in km
        swatVar(i)=swatVar(i)/(hru_km(i))
      endif

      !reset the counters for the loop
      cellUsed = .false.
    enddo

    return
    end
```

SM_GETGW.F

```
      module sm_getgw
!!    ~ ~ ~ Authors ~ ~ ~
!!    Tyler Wible, Masters student
!!    Colorado State University 2012-2014
!!    Comment initials "tcw"
!!
!!    Andre Dozier, PhD student
!!    Colorado State University 2012-2016
!!    Comment initials "aqd"
!!
!!    ~ ~ ~ Purpose ~ ~ ~
!!    This subroutine sets up events for the groundwater portion of the
!!    SWAT-MODFLOW linkage
!!
        use parm
        use modevent
        implicit none
      contains

        subroutine sm_getgwcontr(eventdata)
          class (ieventdata), pointer :: eventdata
          computegw = .false.
        end subroutine sm_getgwcontr

        subroutine sm_getgwnutr(eventdata)
          class (ieventdata), pointer :: eventdata
          computenutr = .true.
        end subroutine sm_getgwnutr

        subroutine sm_getrechloss(eventdata)
          class (ieventdata), pointer :: eventdata
          initializeRCHloss = .false.
        end subroutine sm_getrechloss


      end module sm_getgw
```

SM_GRID2DHRU2D.F

```
      subroutine sm_grid2dhru2D (mfVar2, smVar)
!!    ~ ~ ~ Author ~ ~ ~
!!    Tyler Wible, Masters student
!!    Colorado State University 2012-2014
!!    Comment initials "tcw"
!!
!!    ~ ~ ~ Purpose ~ ~ ~
!!    This subroutine converts 2D MODFLOW-based grids to SM-based disaggregated
HRUs (dhrus)
!!
!!    ~ ~ ~ Variables Used ~ ~ ~
!!    name        |units        |definition
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!    g2d_r       |none         |Array per SM dhru of the row IDs of the grids
!!                |             |which contribute to this dhru
!!    g2d_c       |none         |Array per SM dhru of the column IDs of the grids
!!                |             |which contribute to this dhru
!!    g2d_area    |none         |Array per SM dhru of the percent area of
!!                |             |the grids which contribute to this dhru
!!    g2d_size    |none         |the maximum number of grids which contribute
!!                |             |to a single dhru, used for looping and
!!                |             |dimensioning variables
!!    mfVar2      |unknown      |2D MODFLOW variable (NCOL, NROW)
!!    smVar       |unknown      |SM variable (list of dhrus) to be populated
!!              |              |with the contents of the MODFLOW variable
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!
!!    ~ ~ ~ Variables Modified ~ ~ ~
!!    name        |units          |definition
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!    smVar       |unknown        |SM variable (list of dhrus) now populated
!!              |              |with the contents of the MODFLOW variable
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!
!!    ~ ~ ~ Local Definitions ~ ~ ~
!!    name        |units          |definition
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!    i           |none           |SM dhru loop index
!!    j           |none           |g2d_size loop index
!!    row         |none           |row index for MODFLOW
!!    col         |none           |column index for MODFLOW
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

!!    ~ ~ ~ ~ ~ ~ END SPECIFICATIONS ~ ~ ~ ~ ~ ~

!     Import variables
      use GLOBAL, only: NCOL,NROW !MODFLOW
      use sm_parm, only: dhru, g2d_size, g2d_r, g2d_c, g2d_area !sm linkage
      implicit none

!     Define local variables
      real mfVar2(NCOL,NROW)
      real smVar(dhru)
      integer i, j, row, col

!     Initialize variables
      smVar = 0.

!     Convert MODFLOW-grid variables into SM-dhru variables by multiplying each
!     contributing grid variable by its percent area contributing to each dhru
      do i=1, dhru
        do j=1, g2d_size
          row = g2d_r(i,j)
          col = g2d_c(i,j)
          if(col.ne.0 .and. row.ne.0)then
            smVar(i) = smVar(i) + mfVar2(col,row)*g2d_area(i,j)
          endif
        enddo
      enddo
      return
      end
```

SM_GRID2DHRU3D.F

```
      subroutine sm_grid2dhru3D (mfVar3, smVar, location)
!!    ~ ~ ~ Author ~ ~ ~
!!    Tyler Wible, Masters student
!!    Colorado State University 2012-2014
!!    Comment initials "tcw"
!!
!!    ~ ~ ~ Purpose ~ ~ ~
!!    This subroutine converts 3D MODFLOW-based grids to SM-based disaggregated
HRUs (dhrus)
!!
!!    ~ ~ ~ Variables Used ~ ~ ~
!!    name        |units        |definition
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!    g2d_r       |none         |Array per SM dhru of the row IDs of the grids
!!                |             |which contribute to this dhru
!!    g2d_c       |none         |Array per SM dhru of the column IDs of the grids
!!                |             |which contribute to this dhru
!!    g2d_area    |none         |Array per SM dhru of the percent area of
!!                |             |the grids which contribute to this dhru
!!    g2d_size    |none         |the maximum number of grids which contribute
!!                |             |to a single dhru, used for looping and
!!                |             |dimensioning variables
!!    mfVar3      |unknown      |MODFLOW variable (NCOL, NROW, NLAY)
!!    smVar       |unknown      |SM variable (list of dhru) to be populated
!!                |             |with the contents of the MODFLOW variable
!!    location    |none         |an array(NCOL, NROW) with the value equal to which
!!                |             |layer(NLAY) the MODFLOW value is to be taken from
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!
!!    ~ ~ ~ Variables Modified ~ ~ ~
!!    name        |units        |definition
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!    smVar       |unknown      |SM variable (list of dhrus) now populated
!!                |             |with the contents of the MODFLOW variable
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!
!!    ~ ~ ~ Local Definitions ~ ~ ~
!!    name        |units        |definition
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!    i           |none         |SM dhru loop index
!!    j           |none         |g2d_size loop index
!!    row         |none         |row index for MODFLOW
!!    col         |none         |column index for MODFLOW
!!    lay         |none         |layer index for MODFLOW
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

!!    ~ ~ ~ ~ ~ ~ END SPECIFICATIONS ~ ~ ~ ~ ~ ~

!     Import variables
      use GLOBAL, only: NCOL,NROW,NLAY !MODFLOW
      use sm_parm, only: dhru, g2d_size, g2d_r, g2d_c, g2d_area !sm linkage
      implicit none

!     Define local variables
!     real sum
      double precision mfVar3(NCOL,NROW,NLAY)
      real smVar(dhru)
      real location(NCOL,NROW)
      integer i, j, row, col, lay

!     Initialize variables
!     sum = 0
      smVar = 0.

!     Convert MODFLOW-grid variables into SM-dhru variables by multiplying each
!     contributing grid variable by its percent area contributing to each dhru
!     For 3D variable arrays taking specified layer value only
      do i=1, dhru
        do j=1, g2d_size
          row = g2d_r(i,j)
          col = g2d_c(i,j)
          if(col.ne.0 .and. row.ne.0)then
            lay = location(col,row)
            smVar(i) = smVar(i) +
     &          mfVar3(col,row,lay)*g2d_area(i,j)
          endif
        enddo
      enddo

      return
      end
```

SM_HRU2DHRU.F

```
      subroutine sm_hru2dhru (swatVar, smVar)
!!    ~ ~ ~ Author ~ ~ ~
!!    Tyler Wible, Masters student
!!    Colorado State University 2012-2014
!!    Comment initials "tcw"
!!
!!    ~ ~ ~ Purpose ~ ~ ~
!!    This subroutine converts a SWAT HRU-variable to a SM disaggregated HRUs
(dhrus) variable
!!
!!    ~ ~ ~ Variables Used ~ ~ ~
!!    name        |units        |definition
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!    d2h_id      |none         |Array per SWAT hru of the IDs of the dhrus
!!                |             |which contribute to this hru
!!    swatVar     |unknown      |SWAT variable (list of nhru)
!!    smVar       |unknown      |SM variable (list of dhrus) to be populated
!!                |             |with the contents of the SWAT variable
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!
!!    ~ ~ ~ Variables Modified ~ ~ ~
!!    name        |units        |definition
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!    smVar       |unknown      |SM variable (list of dhrus) to be populated
!!                |             |with the contents of the SWAT variable
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!
!!    ~ ~ ~ Local Definitions ~ ~ ~
!!    name        |units        |definition
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!    i           |none         |SWAT HRU loop index
!!    j           |none         |SM dhru loop index
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

!!    ~ ~ ~ ~ ~ ~ END SPECIFICATIONS ~ ~ ~ ~ ~ ~

!     Import variables
      use parm, only: nhru !SWAT
      use sm_parm, only: dhru, d2h_size, d2h_id, d2h_area !sm linkage
      implicit none

!     Define local variables
      real swatVar(nhru)
      real smVar(dhru)
      integer i, j

!     Initialize variables
      smVar = 0.
```

```
!     Convert SWAT-HRU variables to SM-dhru variables
!     The conversion does not involve a weighted average because the SWAT variables
!     are calculated such that the variable's value is the same within each DHRU
!     The Water Table in SWAT is a depth. The depth is the same everywhere in the HRU.
!     Thus, any dhru will also have the same value. Hence, do not apply a weighted
average.
      do i=1, nhru
        do j=1, d2h_size
          if(d2h_id(i,j).ne.0)then
            smVar(d2h_id(i,j)) = swatVar(i)
          endif
        enddo
      enddo

      return
      end
```

SM_INIT_MF.F

```
      subroutine sm_init_mf(eventdata)
!!    ~ ~ ~ Authors ~ ~ ~
!!    Tyler Wible, Masters student
!!    Colorado State University 2012-2014
!!    Comment initials "tcw"
!!
!!    Andre Dozier, PhD student
!!    Colorado State University 2012-2016
!!    Comment initials "aqd"
!!
!!    Ryan Bailey, Post-Doc student (2012-2013), Assistant Professor (2013-)
!!    Colorado State University 2012-
!!    Comment initials "rtb"
!!
!!    ~ ~ ~ PURPOSE ~ ~ ~
!!    This subroutine initializes MODFLOW and the SWAT-MODFLOW (SM) linking
!!    subroutines
!!
        use modevent
        implicit none
        class (ieventdata), pointer :: eventdata

!     Set up MODFLOW data and allocate arrays
        print *, 'MODFLOW is being used' !rtb
        call mf_read !rtb
        call sm_read_grid2dhru !tcw
        call sm_read_dhru2grid !tcw
        call sm_read_dhru2hru !tcw
        call sm_read_river2grid !tcw
      end subroutine sm_init_mf
```

SM_MAIN.F

```
    program main
!!   ~ ~ ~ Authors ~ ~ ~
!!   Tyler Wible, Masters student
!!   Colorado State University 2012-2014
!!   Comment initials "tcw"
!!
!!   Andre Dozier, PhD student
!!   Colorado State University 2012-2016
!!   Comment initials "aqd"
!!
!!   ~ ~ ~ PURPOSE ~ ~ ~
!!   This subroutine links sets up the SWAT-MODFLOW (SM) linking subroutines
!!   and "events" and then calls SWAT-MODFLOW through "events"
!!
      use sm_parm, only: mf_active
      implicit none

      ! Read the swat-modflow input files
      call sm_read_link !tcw

      ! Subscribe to events within modflow and swat (in order to
      !     essentially wrap the models with connection routines)
      if (mf_active.eq.1) then
        call pkg_modflow
        call pkg_swat
      end if

      ! Run SWAT's main subroutine
      call swat_main
      ! MODFLOW is called within SWAT with Command 19

      ! Close swat-modflow (close files, deallocate variables, etc.)s
      call sm_close
      call mf_close

    end program main
```

SM_MF_READ.F

```
    subroutine sm_mf_read(eventdata)
```

```
!!    ~ ~ ~ Authors ~ ~ ~
!!    Tyler Wible, Masters student
!!    Colorado State University 2012-2014
!!    Comment initials "tcw"
!!
!!    Andre Dozier, PhD student
!!    Colorado State University 2012-2016
!!    Comment initials "aqd"
!!
!!    ~ ~ ~ Purpose ~ ~ ~
!!    This subroutine converts the provided array to different units based on
!!    provided flags for the incoming and outgoing units (based on a modified
!!    set of MODFLOW's unit flags listed below)
!!
!!    ~ ~ ~ Local Variables ~ ~ ~
!!    name        |units         |definition
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!    leapyr      |none          |leap year flag for unit conversions
!!                |              |involving years
!!                |              |0  leap year
!!                |              |1  regular year
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!    ~ ~ ~ ~ ~ ~ END SPECIFICATIONS ~ ~ ~ ~ ~ ~

!     Initialize local variables
      use modevent
      use parm, only: leapyr !SWAT
      use GLOBAL, only: readMFinput, MF_leapyr !MODFLOW
      implicit none
      class (ieventdata), pointer :: eventdata

      !mf_read has already been called by SM linkage, so do not call it again
      readMFinput = .false.

      !make sure modflow's unit conversion for the time-step are based on years or leap
years correctly
      MF_leapyr = leapyr

    end
```

SM_MF_RUN.F

```
    subroutine sm_mf_run(eventdata)
!!    ~ ~ ~ Authors ~ ~ ~
!!    Tyler Wible, Masters student
!!    Colorado State University 2012-2014
!!    Comment initials "tcw"
!!
!!    Andre Dozier, PhD student
!!    Colorado State University 2012-2016
!!    Comment initials "aqd"
!!
!!    ~ ~ ~ PURPOSE ~ ~ ~
!!    This subroutine calls conversions from SWAT to MODFLOW, then runs
MODFLOW
!!    for one day, then converts back the groundwater results from MODFLOW to
!!    SWAT
!!
        use modevent
        implicit none
        class (ieventdata), pointer :: eventdata

        !Convert SWAT units into MODFLOW units
        call sm_conversion2mf

        !water table
        !river discharge/seepage
        !pumping
        call mf_run

        !Convert MODFLOW units back to SWAT units
        call sm_conversion2swat

      end subroutine sm_mf_run
```

SM_MFRIVER.F

```
      subroutine sm_mfRiver(eventdata)
!!    ~ ~ ~ Author ~ ~ ~
!!    Tyler Wible, Masters student
!!    Colorado State University 2012-2014
!!    Comment initials "tcw"
!!
!!    ~ ~ ~ PURPOSE ~ ~ ~
!!    This subroutine converts the necessary SWAT variables used by MODFLOW's river
package into
!!    the required hydrauilc conductivity and stage in MODFLOW one a daily timestep
!!
!!    ~ ~ ~ Variables Used~ ~ ~
!!    name           |units    |definition
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!    LENUNI          |unit_in |the MODFLOW variable for which length units are being
used
!!    dep_chan(:)      |m        |average daily water depth in channel
!!    grid2riv_len    |m        |a list per MODFLOW river grids (cols) containing
!!                    |         |the SWAT river lengths within that grid cell
!!    nriver_cells    |n/a      |the total number of river grid cells in MODFLOW
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!
!!    ~ ~ ~ Variables Modified ~ ~ ~
!!    name            |units   |definition
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!    RIVR(4,:)       |        |river stage (depth + bottom elevation) filled in with info from
SWAT per day
!!    RIVR(5,:)       |        |river conducance, filled in with info from SWAT per day (in
order to overwrite re-reading the .RIVR file every timeset)
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!
!!    ~ ~ ~ Local Definitions ~ ~ ~
!!    name          |units     |definition
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!    mf_lengthUnit |LENUNI    |the modified inteteger to represent
!!                  |          |the MODFLOW unit of length for units
!!    rivlen        |LENUNI    |variable for total river length in current
!!                  |          |river grid cell
!!    rivdepth      |ITMUNI    |variable for weighted average (based on river
!!                  |          |length) for depth of water in current river grid cell
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!
!!    ~ ~ ~ ~ ~ ~ END SPECIFICATIONS ~ ~ ~ ~ ~ ~

      !Import variables
      use modevent
      use parm, only:msub,dep_chan,leapyr !SWAT
      use GLOBAL, only:LENUNI !MODFLOW
      use GWFRIVMODULE, only:RIVR !MODFLOW
      use sm_parm !sm linkage
      implicit none
      class (ieventdata), pointer :: eventdata

      !Define local variables
      integer mf_lengthUnit,mf_timeUnit,i,j,k,subIndex
      real rivlen(1)
      real rivdepth(1)
      mf_lengthUnit = LENUNI + 10
      ! Loop through the SWAT rivers to get channel depth (dep_chan) to pass into
   MODFLOW's RIVR
      do i=1, nriver_cells
        !reset averaged river properties
        rivlen = 0.
        rivdepth = 0.

        !Loop through the SWAT rivers attributes needed to calculate depth
        do j=1, msub
          !Get the river's properties to be based on a weighted average with:
          !weights = subbasin's river segment length / total river length in grid cell (rivlen)
          rivlen(1) = rivlen(1) + grid2riv_len(i,j)
          rivdepth(1) = rivdepth(1)+dep_chan(j)*grid2riv_len(i,j)
        enddo

        if(rivlen(1).eq.0) rivlen(1) = 1. !prevent divide by zero problems
        !Take weighted average of river depth
        rivdepth(1) = rivdepth(1)/rivlen(1)

        !Convert into MODFLOW units
        call units(rivdepth, 12, mf_lengthUnit, 1, 1, leapyr);! to convert length units (m to
   LENUNI)

        !Populate MODFLOW's RIVR variable for this time step's river stage
        RIVR(4,i) = rivdepth(1) + RIVR(6,i)! cell stage (depth + bottom elevation)
      enddo


      return
      end
```

SM_NORMOUT.F

```
      subroutine sm_normOut(eventdata)
   !!   ~ ~ ~ Authors ~ ~ ~
   !!   Tyler Wible, Masters student
   !!   Colorado State University 2012-2014
   !!   Comment initials "tcw"
   !!
   !!   Andre Dozier, PhD student
   !!   Colorado State University 2012-2016
   !!   Comment initials "aqd"
   !!
   !!   ~ ~ ~ PURPOSE ~ ~ ~
   !!   This subroutine updates a flag in SWAT to not save
   !!   groundwater discharge variables (gw_q and gw_qdeep)
```

```
!!    as this is now taken care of in sm_updateOutput.f
!!
        use modevent
        use parm, only: normOut !SWAT
        implicit none
        class (ieventdata), pointer :: eventdata

        !Because SWAT-MODFLOW will update groundwater's output summary variables
itself, turn off the normal summary in sumv.f
        normOut = .false.

      end subroutine
```

SM_PARM.F

```
      module sm_parm
!!    ~ ~ ~ Author ~ ~ ~
!!    Tyler Wible, Masters student
!!    Colorado State University 2012-2014
!!    Comment initials "tcw"
!!
!!    ~ ~ ~ PURPOSE ~ ~ ~
!!    This module contains variables used for linking SWAT variables to MODFLOW and
!!    MODFLOW variables to SWAT
!!
!!    Declare global variables for SWAT-MODFLOW (SM) linkage
      real, dimension (:,:), allocatable :: g2d_r, g2d_c, g2d_area
      real, dimension (:,:), allocatable :: d2g_id, d2g_area
      real, dimension (:,:), allocatable :: d2h_id, d2h_area
      integer :: g2d_size, d2g_size, d2h_size
      real, dimension (:,:), allocatable :: grid2riv_id, grid2riv_len

!!    Declare flag for whether modflow is active
      integer :: mf_active

!!    SM Disaggregated HRU (dhru) variables
      real, dimension (:), allocatable :: etremain_dhru, sepbtm_dhru
      integer :: dhru

!!    MODFLOW/SWAT River segment variables
      integer :: nriver_cells

      end module sm_parm
```

SM_READ_DHRU2GRID.F

```
      subroutine sm_read_dhru2grid
!!    ~ ~ ~ Author ~ ~ ~
!!    Tyler Wible, Masters student
!!    Colorado State University 2012-2014
!!    Comment initials "tcw"
!!
!!    ~ ~ ~ Purpose ~ ~ ~
!!    This subroutine reads in the file containing the information to convert
!!    SM-based disaggregated hrus (dhru) variables to MODFLOW-based grid variables
!!
!!    ~ ~ ~ Variables Used ~ ~ ~
!!    name        |units        |definition
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!    NCOL        |none         |the current number of columns of MODFLOW grids
!!    NROW        |none         |the current number of rowss of MODFLOW grids
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!
!!    ~ ~ ~ Variables Modified ~ ~ ~
!!    name        |units        |definition
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!    d2g_id      |none         |Array per MODFLOW grid of the IDs of the dhrus
!!                |             |which contribute to this grid
!!    d2g_area    |none         |Array per MODFLOW grid of the percent area of
!!                |             |the dhrus which contribute to this grid
!!    d2g_size    |none         |the maximum number of dhrus which contribute
!!                |             |to a single grid, used for looping and
!!                |             |dimensioning variables
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!
!!    ~ ~ ~ Local Definitions ~ ~ ~
!!    name        |units        |definition
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!    i           |none         |counter index for the number of MOFLOW grids
!!    j           |none         |counter index for reading in contributing
!!                |             |areas and looping through all hrus
!!    temp        |none         |index of current MODFLOW grid
!!    nhru_current |none        |index of the number of contributing areas
!!                |             |to loop through
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!    ~ ~ ~ ~ ~ ~ END SPECIFICATIONS ~ ~ ~ ~ ~ ~

!     Import variables
      use GLOBAL, only: NCOL, NROW, NLAY !MODFLOW
      use sm_parm, only: d2g_size, d2g_id, d2g_area !sm linkage
      implicit none

!     Initialize local variables
      integer i, j, gridID, nhru_current, numGrid

!     The first line of this file is the total number of MODFLOW grids (active and inactive)
= NROW * NCOL
      open (5005,file="map_dhru2grid.txt")
      read(5005,*) numGrid, d2g_size

!     Initialize variables
      allocate(d2g_id(NCOL*NROW, d2g_size))
      allocate(d2g_area(NCOL*NROW, d2g_size))
      d2g_id = 0.
      d2g_area = 0.

      do i=1, numGrid
        read(5005,*) gridID, nhru_current ! grid # then the number of dhrus contributing to
this grid cell

        if(nhru_current.gt.0)then
          read(5005,*) (d2g_id(gridID,j),j=1,nhru_current) ! list of dhru ID numbers which
contribute to this grid cell
          read(5005,*) (d2g_area(gridID,j),j=1,nhru_current) ! list of % areas of that dhru
contributing to this grid cell
        else
          read(5005,*)
          read(5005,*)
        endif

      enddo
      close(5005)

      return
      end
```

SM_READ_DHRU2HRU.F

```
      subroutine sm_read_dhru2hru
!!    ~ ~ ~ Author ~ ~ ~
!!    Tyler Wible, Masters student
!!    Colorado State University 2012-2014
!!    Comment initials "tcw"
!!
!!    ~ ~ ~ Purpose ~ ~ ~
!!    This subroutine reads in the file containing the information to convert
!!    SM-based disaggregated HRU (dhru) variables to SWAT-based normal/aggregated
HRU variables
!!
!!    ~ ~ ~ Variables Used ~ ~ ~
!!    name          |units          |definition
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!    nhru          |none           |the current number of SWAT hrus
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!
!!    ~ ~ ~ Variables Modified ~ ~ ~
!!    name          |units          |definition
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!    d2h_id        |none           |Array per SWAT hru of the IDs of the dhrus
!!                  |               |which contribute to this hru
!!    d2h_area      |none           |Array per SWAT hru of the percent area of
!!                  |               |the dhrus which contribute to this hru
!!    d2h_size      |none           |the maximum number of dhrus which contribute
!!                  |               |to a single hru, used for looping and
!!                  |               |dimensioning variables
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!
!!    ~ ~ ~ Local Definitions ~ ~ ~
!!    name          |units          |definition
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!    i             |none           |counter index for reading in contributing
!!                  |               |areas and looping through all grids
!!    j             |none           |counter index for the number of SM dhru
!!    hruID         |none           |index of current SM dhru
!!    dhru_current  |none           |index of the number of contributing areas
!!                  |               |to loop through
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!    ~ ~ ~ ~ ~ ~ END SPECIFICATIONS ~ ~ ~ ~ ~ ~

!     Import variables
      use sm_parm, only: d2h_size, d2h_id, d2h_area !sm linkage
      implicit none

!     Initialize local variables
      integer ahru,i,j,hruID,dhru_current,subID
      real, dimension (:,:), allocatable :: dhrulist
      real blah

!     Read in the ID and percent area of each SM dhru contributing to each SWAT HRU
      open (5006,file="map_dhru2hru.txt")
```

```
!     The first line of this file is the total number of HRUs in the watershed (aka, how
many will be read in)
      read(5006,*) ahru, d2h_size !must equal SWAT's nhru

!     Initialize variables
      allocate(d2h_id(ahru, d2h_size))
      allocate(d2h_area(ahru, d2h_size))
      d2h_id = 0.
      d2h_area = 0.

      do i=1, ahru
        ! the HRU's global ID within the watershed, the number of dhrus contributing to this
HRU,
        ! the subbasin number for this HRU, the river ID that this HRU drains to, the
segement ID of
        ! the river that this HRU drains to, the strahlor stream order of this river
        read(5006,*) hruID, dhru_current, subID

        if(dhru_current.gt.0)then
          read(5006,*) (d2h_id(hruID,j),j=1,dhru_current) ! list of dhru ID numbers which
contribute to this hru
          read(5006,*) (d2h_area(hruID,j),j=1,dhru_current) ! list of % areas of that dhru
contributing to this hru
        else
          read(5006,*)
          read(5006,*)
        endif

      enddo
      close(5006)

      return
      end
```

SM_READ_GRID2DHRU.F

```
      subroutine sm_read_grid2dhru
!!    ~ ~ ~ Author ~ ~ ~
!!    Tyler Wible, Masters student
!!    Colorado State University 2012-2014
!!    Comment initials "tcw"
!!
!!    ~ ~ ~ Purpose ~ ~ ~
!!    This subroutine reads in the file containing the information to convert
!!    MODFLOW-based grid variables to SM-based disaggregated HRU (dhru) variables
!!
```

```
!!    ~ ~ ~ Variables Used ~ ~ ~
!!    name         |units         |definition
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!    dhru         |none          |the total number of disaggregated HRUs in the
!!                 |              |entire watershed (not just the current subbasin)
!!    NCOL         |none          |the current number of columns of MODFLOW grids
!!    NROW         |none          |the current number of rowss of MODFLOW grids
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!
!!    ~ ~ ~ Variables Modified ~ ~ ~
!!    name         |units         |definition
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!    g2d_r        |none          |Array per SM dhru of the row IDs of the grids
!!                 |              |which contribute to this dhru
!!    g2d_c        |none          |Array per SM dhru of the column IDs of the grids
!!                 |              |which contribute to this dhru
!!    g2d_area     |none          |Array per SM dhru of the percent area of
!!                 |              |the grids which contribute to this dhru
!!    g2d_size     |none          |the maximum number of grids which contribute
!!                 |              |to a single dhru, used for looping and
!!                 |              |dimensioning variables
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!
!!    ~ ~ ~ Local Definitions ~ ~ ~
!!    name         |units         |definition
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!    i            |none          |counter index for the number of SM dhru
!!    j            |none          |counter index for reading in contributing
!!                 |              |areas and looping through all grids
!!    dhruID       |none          |index of current SM dhru
!!    ngrid_current|none          |index of the number of contributing areas
!!                 |              |to loop through
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!    ~ ~ ~ ~ ~ ~ END SPECIFICATIONS ~ ~ ~ ~ ~ ~

!     Import variables
      use GLOBAL, only: NCOL, NROW, NLAY !MODFLOW
      use sm_parm, only: dhru, g2d_size, g2d_r, g2d_c, g2d_area !sm linkage
      implicit none

!     Initialize local variables
      integer i, j, dhruID, ngrid_current
      real, dimension (:,:), allocatable :: gridlist

!     Read in the ID and percent area of each MODFLOW grid contributing to each SM
dhru
```

```
      open (5004,file="map_grid2dhru.txt")

!     The first line of this file is the total number of disaggregated hrus in the watershed
(aka, how many will be read in)
      read(5004,*) dhru, g2d_size

!     Initialize variables
      allocate(g2d_r(dhru, g2d_size))
      allocate(g2d_c(dhru, g2d_size))
      allocate(g2d_area(dhru, g2d_size))
      g2d_r = 0.
      g2d_c = 0.
      g2d_area = 0.

      do i=1, dhru
        read(5004,*) dhruID, ngrid_current ! the dhru's global ID within the watershed, the
number of grids contributing to this dhru

        if(ngrid_current.gt.0)then
          read(5004,*) (g2d_r(dhruID,j),j=1,ngrid_current) ! list of grid row ID numbers
which contribute to this dhru
          read(5004,*) (g2d_c(dhruID,j),j=1,ngrid_current) ! list of grid column ID numbers
which contribute to this dhru
          read(5004,*) (g2d_area(dhruID,j),j=1,ngrid_current) ! list of % areas of that grid
contributing to this dhru
        else
          read(5004,*)
          read(5004,*)
          read(5004,*)
        endif

      enddo
      close(5004)

      return
      end
```

SM_READ_LINK.F

```
      subroutine sm_read_link
!!    ~ ~ ~ Author ~ ~ ~
!!    Tyler Wible, Masters student
!!    Colorado State University 2012-2014
!!    Comment initials "tcw"
!!
!!    ~ ~ ~ Purpose ~ ~ ~
!!    This subroutine reads in the file containing the information to linke
!!    SWAT and MODFLOW
!!
!!    ~ ~ ~ Variables Modified ~ ~ ~
!!    name          |units          |definition
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!    mf_active     |none           |index whether or not to use MODFLOW to
!!                  |               |calculate groundwater flow processes
!!                  |               |instead of SWAT's gwmod
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!    ~ ~ ~ ~ ~ ~ END SPECIFICATIONS ~ ~ ~ ~ ~ ~

!     Import variables
      use sm_parm, only: mf_active !SM linkage
      implicit none

!     Read in extra information for linking SWAT and MODFLOW
      open(5003,file="sm_link.txt")
      read(5003, '(I20)') mf_active
      close(5003)

      return
      end
```

SM_READ_RIVER2GRID.F

```
      subroutine sm_read_river2grid()
!!    ~ ~ ~ Author ~ ~ ~
!!    Tyler Wible, Masters student
!!    Colorado State University 2012-2014
!!    Comment initials "tcw"
!!
!!    ~ ~ ~ Purpose ~ ~ ~
!!    This subroutine reads in the file containing the information to calculate
!!    river conductance for MODFLOW's river cells
!!
!!    ~ ~ ~ Variables Used ~ ~ ~
!!    name          |units          |definition
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!    subtot        |none           |number of subbasins in watershed
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!
!!    ~ ~ ~ Variables Modified ~ ~ ~
!!    name          |units          |definition
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!    grid2riv_id   |none           |a list per MODFLOW river grids (cols) containing
!!              |            |the SWAT river IDs within that grid cell
!!   grid2riv_len  |m            |a list per MODFLOW river grids (cols) containing
!!              |            |the SWAT river lengths within that grid cell
!!   river_cells   |LENUNI       |an array of the properties of the river grid cells,
!!              |            |are the thicknesses of the river bed of the grid cells
!!   nriver_cells  |n/a          |the total number of river grid cells in MODFLOW
!!   ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!
!!   ~ ~ ~ Local Definitions ~ ~ ~
!!   name          |units         |definition
!!   ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!   i             |none          |counter index for the number of SWAT subbasins
!!   j             |none          |counter index for reading contributing
!!              |            |MODFLOW rive grids
!!   temp          |none          |index of current SWAT subbasin
!!   ngrid_current |none          |index of the number of contributing MODFLOW
!!              |            |river grids
!!   ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!   ~ ~ ~ ~ ~ ~ END SPECIFICATIONS ~ ~ ~ ~ ~ ~

!    Import variables
      use GLOBAL, only:LENUNI,ITMUNI !MODFLOW
      use sm_parm !sm linkage
      implicit none

!    Initialize local variables
      integer i, j, temp, nriv_current

!    Read in the id and percent area of each SWAT HRU contributing to each
!    MODFLOW grid
      open (5007,file="map_river2grid.txt")
      read(5007,*) nriver_cells ! the total # of river cells in MODFLOW

!    Allocate the variable sizes, which needs to be called here before reading the file
continues
      call sm_allocate()

!    Read which SWAT river reaches are within each MODFLOW river cell, and their
associated river lengths
      do i=1, nriver_cells
        read(5007,*) temp, nriv_current ! grid #, then the number of river segments within
this grid
        if(nriv_current.gt.0)then
          read(5007,*) (grid2riv_id(i,j), j=1,nriv_current) ! list of river ID numbers which are
within the current grid
```

```
          read(5007,*) (grid2riv_len(i,j),j=1,nriv_current) ! list of length of river within the
current grid
        else
          read(5007,*)
          read(5007,*)
        endif

      enddo
      close(5007)

      return
      end
```

SM_RECHARGE.F

```
      subroutine sm_recharge(eventdata)
!!    ~ ~ ~ Authors ~ ~ ~
!!    Tyler Wible, Masters student
!!    Colorado State University 2012-2014
!!    Comment initials "tcw"
!!
!!    Andre Dozier, PhD student
!!    Colorado State University 2012-2016
!!    Comment initials "aqd"
!!
!!    ~ ~ ~ PURPOSE ~ ~ ~
!!    This subroutine converts the necessary SWAT variables for the MODFLOW
recharge
!!    (RCH) package's variables
!!
        use modevent
        use sm_parm, only: sepbtm_dhru
        use GWFRCHMODULE, only: RECH
        implicit none
        class (ieventdata), pointer :: eventdata

        call sm_dhru2grid2D(sepbtm_dhru,RECH,1)

      end subroutine sm_recharge
```

SM_UPDATEOUTPUT.F

```
      subroutine sm_updateOutput
!!     ~ ~ ~ Author ~ ~ ~
!!     Tyler Wible, Masters student
!!     Colorado State University 2012-2014
!!      Comment initials "tcw"
!!
!!      ~ ~ ~ Purpose ~ ~ ~
!!      This file is temporary only, please delete it later
!!      This file update the SWAT output variables with corrected
!!      groundwater flow values from MODFLOW

!       Import variables
        use parm !SWAT
        implicit none

        integer :: subID,h, k

        !Update SWAT's output variables to reflect the changes to gw_q and gw_qdeep
from MODFLOW, like it is done in sumv.f
        do subID=1, msub
          h = hru1(subID)
          do k=1, hrutot(subID)
            !the below code is borrowed from SWAT's sumv.f
            if (curyr > nyskip) then
              !! HRU summations
              hrumono(6,h) = hrumono(6,h) + gw_q(h)
              hrumono(70,h) = hrumono(70,h) + gw_qdeep(h)

              !! watershed summations
              if (ffcst == 0 .and. iscen == 1) then
                wshddayo(104) = wshddayo(104) + gw_q(h) * hru_dafr(h)
                wshddayo(113) = wshddayo(113) + gw_qdeep(h) *hru_dafr(h)
              else if (ffcst == 1) then
                fcstaao(8) = fcstaao(8) + gw_q(h) * hru_dafr(h)
              end if
            end if
            !end borrowed SWAT code from sumv.f
            h = h + 1
          enddo
        enddo

      return
      end
```

SM_UPFLUXTOSOIL.F

```
      subroutine sm_upflux_to_soil
!!    ~ ~ ~ Authors ~ ~ ~
!!    Tyler Wible, Masters student
!!    Colorado State University 2012-2014
!!   Comment initials "tcw"
!!
!!   Ryan Bailey, Post-Doc student (2012-2013), Assistant Professor (2013-)
!!   Colorado State University 2012-
!!   Comment initials "rtb"
!!
!!   ~ ~ ~ Purpose ~ ~ ~
!!   Add upflux water (calculated by UZF) to soil profile (SWAT)
!!   Start with the bottom soil layer. When filled to Field Capacity, move to
!!   next layer up.
!!
!!   ~ ~ ~ Variables Used ~ ~ ~
!!   name          |units             |definition
!!   ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!   sol_nly(:)    |none              |number of soil layers
!!   sol_z(:,:)    |mm soil           |depth to bottom of soil layer
!!   sol_st(:,:)   |mm H2O            |amount of water stored in the soil
!!                 |                  |layer on any given day (less wp water)
!!   sol_up(:,:)   |mm H2O/mm soil    |water content of soil at -0.033 MPa
!!                 |                  |(field capacity)
!!   sol_sw(:)     |mm H2O            |amount of water stored in soil profile
!!                 |                  |on any given day
!!   sol_wpmm(:,:) |mm H20            |water content of soil at -1.5 MPa
!!                 |                  |(wilting point)
!!   nhru          |none              |number of HRUs in watershed
!!   LENUNI        |unit_in           |the MODFLOW variable for which length
!!                 |                  |units are being used
!!   ITMUNI        |unit_out          |the MODFLOW variable for which time
!!                 |                  |units are being used
!!   NCOL          |n/a               |the MODFLOW variable for number of
!!                 |                  |columns of grids
!!   NROW          |n/a               |the MODFLOW variable for number of
!!                 |                  |rows of grids
!!   NLAY          |n/a               |the MODFLOW variable for number of
!!                 |                  |layers of grids
!!   GWET(:,:,:)   |LENUNI**3/ITMUNI  |a modflow variable for the et coming
!!                 |                  |from the groundwater UZF package
!!                 |                  |(only the UZF or EVT package is allowed
!!                 |                  |active, not both)
!!   UZET(:,:,:)   |LENUNI**3/ITMUNI  |an added modflow variable for the et
!!                 |                  |coming from the groundwater UZF package
!!                 |                  |(only the UZF or EVT package is allowed
!!                 |                  |active, not both)
!!   EVTvol(:,:,:) |LENUNI**3/ITMUNI  |an added modflow variable for the et
!!                 |                  |coming from the groundwater EVT package
!!                 |                  |(only the UZF or EVT package is allowed
!!               |                 |active, not both)
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!
!!    ~ ~ ~ Variables Modified ~ ~ ~
!!    name          |units             |definition
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!    sol_st(:,:)   |mm H2O            |amount of water stored in the soil layer
!!               |                 |on any given day (less wp water)
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!
!!    ~ ~ ~ Local Definitions ~ ~ ~
!!    name          |units               |definition
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!    upflux(:)     |mm H20              |depth of upflux water (as calculated
!!               |                   |from GWET)
!!    dg            |mm soil             |thickness of soil layer
!!    sol_upmm      |mm H2O              |amount of water at field capacity
!!    j             |none                |HRU number
!!    sol_water     |mm H20              |current amount of water in the soil layer
!!    upflux_mm     |mm H20              |depth of upflux water for the current HRU
!!    sol_deficit   |?                   |the amount of water that can potentially
!!               |                   |be added to the soil layer (based on field capacity)
!!    fract_upflux  |none                |determine fraction of total upflux water
!!               |                   |that is directed to the current layer
!!    no3mass_add   |??                  |based on fract_upflux this is how much of the no3 mass
!!               |                   |should be added to the layer
!!    mf_lengthUnit |MODFLOW length unit |the modified inteteger to represent
!!               |                   |the MODFLOW unit of length
!!    mf_timeUnit   |MODFLOW time unit   |the modified integer to represent
!!               |                   |the MODFLOW unit of time
!!    mf_et(:,:)    |LENUNI**3/ITMUNI    |variable to contain each MODFLOW
!!               |                   |grid cell's et to pass to SWAT
!!               |                   |= GWET
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!    ~ ~ ~ ~ ~ ~ END SPECIFICATIONS ~ ~ ~ ~ ~ ~

!     Import variables
      use parm, only: sol_nly,sol_z,sol_st,sol_up,sol_sw,sol_wpmm, !SWAT
     &                nhru, leapyr
      use GLOBAL, only:LENUNI,ITMUNI,NCOL,NROW,NLAY,DELR,DELC,IUNIT
!MODFLOW
      use GWFUZFMODULE, only: GWET, UZET !MODFLOW UZF package
      USE GWFEVTMODULE,ONLY: EVTvol !MODFLOW EVT package
      use sm_parm, only: dhru !sm linkage
      implicit none
```

```
!     Initialize local variables
      integer i,j,k,ly,mf_lengthUnit,mf_timeUnit,nlayers,layer
      real dg,upflux_mm,sol_upmm,sol_water,sol_deficit,fract_upflux
      real mf_et(NCOL,NROW)
      real upflux_dhru(dhru)
      real upflux(nhru)
      mf_et = 0.
      mf_lengthUnit = LENUNI + 10
      mf_timeUnit = ITMUNI


      !Calculate the total ET upflux from MODFLOW from its uzf groundwater ET
(GWET) variable for each grid cell
      do j=1, NROW
        do i=1, NCOL
          do k=1, NLAY
            !If using the EVT package, get ET from there
            if(IUNIT(5).gt.0)then
              mf_et(i,j) = mf_et(i,j) + EVTvol(i,j,k)/(DELR(i)*DELC(j))
            endif

            !If using the UZF package, get ET from there
            if(IUNIT(55).gt.0)then
              mf_et(i,j) = mf_et(i,j) + UZET(i,j,k)/(DELR(i)*DELC(j))
            endif
          enddo


          !If using the UZF package, get addtional ET from there
          if(IUNIT(55).gt.0)then
            mf_et(i,j) = mf_et(i,j) + GWET(i,j)/(DELR(i)*DELC(j))
          endif
       enddo
      enddo

!     Convert MODFLOW variable into SM variable
      call sm_grid2dhru2D(mf_et, upflux_dhru)! Map the MODFLOW upflux from grids
to dhrus


!     Convert SM variable into SWAT units
      call units(upflux_dhru, mf_lengthUnit, 15, 3, dhru, leapyr)! to convert length units
(LENUNI**3 to km**3)
      call units(upflux_dhru, mf_timeUnit, 4, 1, dhru, leapyr)! to convert time units
(ITMUNI to days)
      call units(upflux_dhru, 15, 14, 1, dhru, leapyr)! to convert length units (km to mm)
```

```
!     Convert SM variable into SWAT variable
      call sm_dhru2hru(upflux_dhru, upflux, 2)! Map the MODFLOW upflux from dhrus
to HRUs and divide the SWAT variable by hru area (km**3/km**2 = km)


      do j=1, nhru
        !upflux_mm: the depth of upflux for the HRU (calculated by dividing the flow rate
of (UZET + GWET) by the area of the HRU)
        upflux_mm = upflux(j) !get the upflux for the HRU

        !add upflux water to the soil layers - beginning with the bottom layer ------------------
----

        !loop through the soil layers (beginning with the bottom layer)
        nlayers = sol_nly(j)
        do k=1,nlayers

          !only proceed if there is any upflux water remaining
          if(upflux_mm.gt.0) then
            layer = nlayers - (k-1)

            !calculate thickness of soil layer
            if(layer.gt.1) then
              dg = sol_z(layer,j) - sol_z(layer-1,j)
            else
              dg = sol_z(layer,j)
            endif

            !determine total water (mm) at Field Capacity (what the soil layer can hold)
            sol_upmm = sol_up(layer,j) * dg !(mm water / mm soil) * mm soil

            !determine the current amount of water in the soil layer
            !(must add wilting point water, since sol_st does not include it)
            sol_water = sol_st(layer,j) + sol_wpmm(layer,j)

            !calculate amount of water that can potentially be added to the soil layer
            !(based on field capacity)
            sol_deficit = sol_upmm - sol_water

            !calculate how much upflux water is added to the soil layer
            if(upflux_mm.ge.sol_deficit) then !Fill up all of the soil layer

              !fill the soil layer to field capacity (then subtract the wilting point
              !water, since this is part of the definition of sol_st)
            sol_st(layer,j)=(sol_water+sol_deficit)-sol_wpmm(layer,j)

            !decrease the amount of upflux that can be added to the above layers
            !(the next time through the loop)
            upflux_mm = upflux_mm - sol_deficit

          else !Fill up part of the soil layer
            !add all of the upflux (resulting water amount in soil layer should be less
            !than field capacity)
            sol_st(layer,j)=(sol_water + upflux_mm)-sol_wpmm(layer,j)
            upflux_mm = 0

          endif

        endif
      enddo !go to the above layer

      !update total soil water amount for the profile ---------------------------------------------
      sol_sw(j) = 0.
      do ly = 1, sol_nly(j)
        sol_sw(j) = sol_sw(j) + sol_st(ly,j)
      enddo
    enddo

    return
    end
```

SM_UZF.F

```
      subroutine sm_uzf(eventdata)
!!    ~ ~ ~ Authors ~ ~ ~
!!    Tyler Wible, Masters student
!!    Colorado State University 2012-2014
!!    Comment initials "tcw"
!!
!!    Andre Dozier, PhD student
!!    Colorado State University 2012-2016
!!    Comment initials "aqd"
!!
!!    ~ ~ ~ PURPOSE ~ ~ ~
!!    This subroutine converts the necessary SWAT variables for the MODFLOW
!!    unsaturated zone flow (UZF1) package's variables
!!
        use modevent
        ! Inside of MODFLOW, infiltration and ET is
        ! provided by SWAT
```

```
  use sm_parm, only: sepbtm_dhru, etremain_dhru
  use GWFUZFMODULE, only: FINF, PETRATE
  implicit none
  class (ieventdata), pointer :: eventdata

  call sm_dhru2grid2D(sepbtm_dhru,FINF,1)
  call sm_dhru2grid2D(etremain_dhru, PETRATE,1)

end subroutine
```

UNITS.F

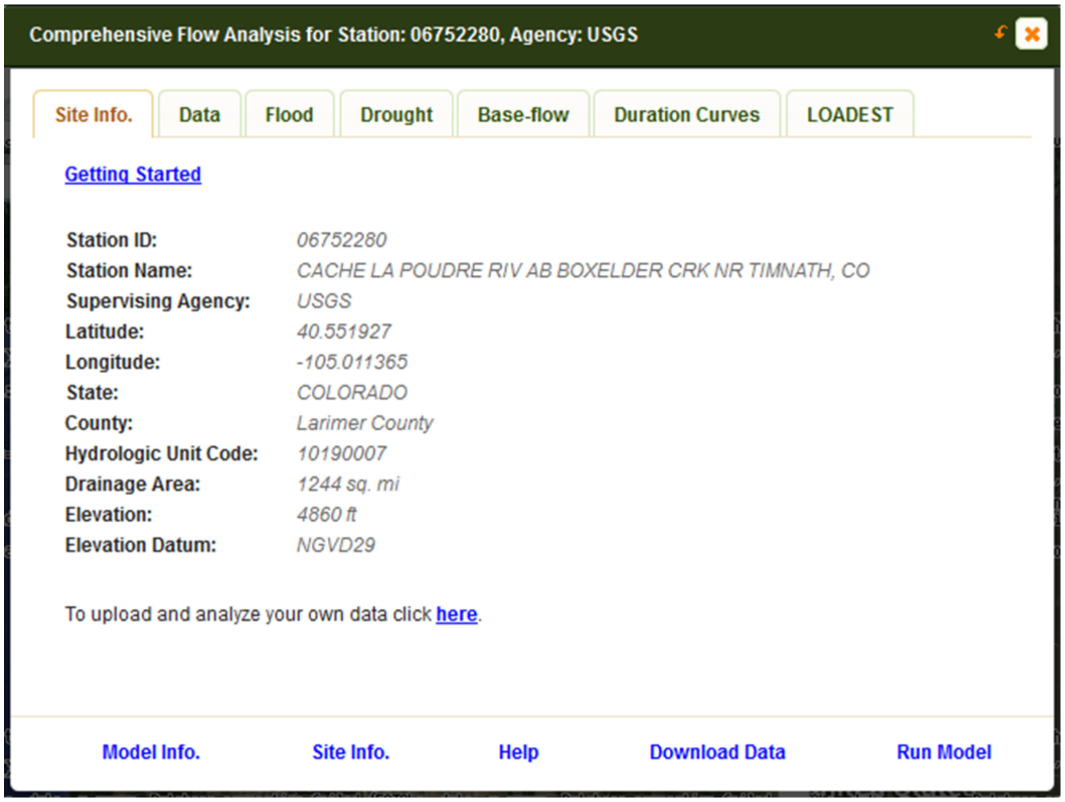
```
      subroutine units(array,unit_in,unit_out,magnitude,asize,leapyr)
!!    ~ ~ ~ Author ~ ~ ~
!!    Tyler Wible, Masters student
!!    Colorado State University 2012-2014
!!    Comment initials "tcw"
!!
!!    ~ ~ ~ Purpose ~ ~ ~
!!    This subroutine converts the provided array to different units based on
!!    provided flags for the incoming and outgoing units (based on a modified
!!    set of MODFLOW's unit flags listed below)
!!
!!    0 = undefined
!!    1 = seconds
!!    2 = minutes
!!    3 = hours
!!    4 = days
!!    5 = years    Note: a year is assumed to be equal to 365 days
!!
!!    11 = feet
!!    12 = meters
!!    13 = centimeters
!!    14 = millimeters
!!    15 = kilometers
!!
!!    21 = kilograms
!!    22 = grams
!!    23 = milligrams
!!    24 = micrograms
!!
!!    31 = square feet
!!    32 = square meters
!!    33 = hectacre (ha)
!!
!!    ~ ~ ~ Variables Used ~ ~ ~
!!    name        |units        |definition
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!    array(:)    |unit_in units |the variable to have its units converted
!!                |              |currently in the units of unit_in that has
!!                |              |"asize" elements in the array
!!    unit_in     |none          |integer representing the incoming unit of the
!!                |              |parameter following the codes listed above
!!    unit_out    |none          |integer representing the outgoing unit of the
!!                |              |parameter following the codes listed above
!!    magnitude   |none          |the exponent of the units to be converted
!!                |              |example: if converting from square feet (ft^2)
!!                |              |to square meeters (m^2), magnitude should have
!!                |              |a value of 2
!!    asize       |              |the number of elements in array(:)
!!    leapyr      |none          |leap year flag, this is only used if the in/out
!!                |              |units are time units involving years
!!                |              |0  leap year
!!                |              |1  regular year
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!
!!    ~ ~ ~ Variables Modified ~ ~ ~
!!    name        |units        |definition
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!    array       |unit_out units|the originally provided variable array now
!!                |              |in the units of unit_out
!!    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
!!    ~ ~ ~ ~ ~ ~ END SPECIFICATIONS ~ ~ ~ ~ ~ ~

!     Initialize local variables
      implicit none
      integer asize, leapyr
      real array(asize)
      real conversion
      integer unit_in, unit_out, magnitude, i
      conversion = 1.

!     Determine the time unit conversion
      if (unit_in.eq.1 .and. unit_out.eq.2) then
        conversion = 1./60.        !!Convert seconds to minutes
      else if (unit_in.eq.1 .and. unit_out.eq.3) then
        conversion = 1./3600.      !!Convert seconds to hours
      else if (unit_in.eq.1 .and. unit_out.eq.4) then
        conversion = 1./86400.     !!Convert seconds to days
      else if (unit_in.eq.1 .and. unit_out.eq.5 .and. leapyr.ne.0) then
        conversion = 1./31536000.  !!Convert seconds to non-leap years
      else if (unit_in.eq.1 .and. unit_out.eq.5 .and. leapyr.eq.0) then
```

```
  conversion = 1./31622400.  !!Convert seconds to leap years

else if (unit_in.eq.2 .and. unit_out.eq.1) then
  conversion = 60.           !!Convert minutes to seconds
else if (unit_in.eq.2 .and. unit_out.eq.3) then
  conversion = 1./60.        !!Convert minutes to hours
else if (unit_in.eq.2 .and. unit_out.eq.4) then
  conversion = 1./1440.      !!Convert minutes to days
else if (unit_in.eq.2 .and. unit_out.eq.5 .and. leapyr.ne.0) then
  conversion = 1./525600.    !!Convert minutes to non-leap years
else if (unit_in.eq.2 .and. unit_out.eq.5 .and. leapyr.eq.0) then
  conversion = 1./527040.    !!Convert minutes to leap years

else if (unit_in.eq.3 .and. unit_out.eq.1) then
  conversion = 3600.         !!Convert hours to seconds
else if (unit_in.eq.3 .and. unit_out.eq.2) then
  conversion = 60.           !!Convert hours to minutes
else if (unit_in.eq.3 .and. unit_out.eq.4) then
  conversion = 1./24.        !!Convert hours to days
else if (unit_in.eq.3 .and. unit_out.eq.5 .and. leapyr.ne.0) then
  conversion = 1./8760.      !!Convert hours to non-leap years
else if (unit_in.eq.3 .and. unit_out.eq.5 .and. leapyr.eq.0) then
  conversion = 1./8784.      !!Convert hours to leap years

else if (unit_in.eq.4 .and. unit_out.eq.1) then
  conversion = 86400.        !!Convert days to seconds
else if (unit_in.eq.4 .and. unit_out.eq.2) then
  conversion = 1440.         !!Convert days to minutes
else if (unit_in.eq.4 .and. unit_out.eq.3) then
  conversion = 24.           !!Convert days to hours
else if (unit_in.eq.4 .and. unit_out.eq.5 .and. leapyr.ne.0) then
  conversion = 1./365.       !!Convert days to non-leap years
else if (unit_in.eq.4 .and. unit_out.eq.5 .and. leapyr.eq.0) then
  conversion = 1./366.       !!Convert days to leap years

else if (unit_in.eq.5 .and. unit_out.eq.1 .and. leapyr.ne.0) then
  conversion = 31536000.     !!Convert non-leap years to seconds
else if (unit_in.eq.5 .and. unit_out.eq.2 .and. leapyr.ne.0) then
  conversion = 525600.       !!Convert non-leap years to minutes
else if (unit_in.eq.5 .and. unit_out.eq.3 .and. leapyr.ne.0) then
  conversion = 8760.         !!Convert non-leap years to hours
else if (unit_in.eq.5 .and. unit_out.eq.4 .and. leapyr.ne.0) then
  conversion = 365.          !!Convert non-leap years days

else if (unit_in.eq.5 .and. unit_out.eq.1 .and. leapyr.eq.0) then
  conversion = 31622400.     !!Convert leap years to seconds
```

```
      else if (unit_in.eq.5 .and. unit_out.eq.2 .and. leapyr.eq.0) then
        conversion = 527040.      !!Convert leap years to minutes
      else if (unit_in.eq.5 .and. unit_out.eq.3 .and. leapyr.eq.0) then
        conversion = 8784.        !!Convert leap years to hours
      else if (unit_in.eq.5 .and. unit_out.eq.4 .and. leapyr.eq.0) then
        conversion = 366.         !!Convert leap years days



!     Determine the distance unit conversion
      else if (unit_in.eq.11 .and. unit_out.eq.12) then
        conversion = 1./3.28084    !!Convert feet to meters
      else if (unit_in.eq.11 .and. unit_out.eq.13) then
        conversion = 1./0.0328084  !!Convert feet to centimeters
      else if (unit_in.eq.11 .and. unit_out.eq.14) then
        conversion = 1./0.00328084 !!Convert feet to millimeters
      else if (unit_in.eq.11 .and. unit_out.eq.15) then
        conversion = 1./3280.84    !!Convert feet to kilometers

      else if (unit_in.eq.12 .and. unit_out.eq.11) then
        conversion = 3.28084      !!Convert meters to feet
      else if (unit_in.eq.12 .and. unit_out.eq.13) then
        conversion = 100.         !!Convert meters to centimeters
      else if (unit_in.eq.12 .and. unit_out.eq.14) then
        conversion = 1000.        !!Convert meters to millimeters
      else if (unit_in.eq.12 .and. unit_out.eq.15) then
        conversion = 1./1000.     !!Convert meters to kilometers

      else if (unit_in.eq.13 .and. unit_out.eq.11) then
        conversion = 0.0328084     !!Convert centimeters to feet
      else if (unit_in.eq.13 .and. unit_out.eq.12) then
        conversion = 1./100.       !!Convert centimeters to meters
      else if (unit_in.eq.13 .and. unit_out.eq.14) then
        conversion = 10.           !!Convert centimeters to millimeters
      else if (unit_in.eq.13 .and. unit_out.eq.15) then
        conversion = 1./100000.    !!Convert centimeters to kilometers

      else if (unit_in.eq.14 .and. unit_out.eq.11) then
        conversion = 0.00328084    !!Convert millimeters to feet
      else if (unit_in.eq.14 .and. unit_out.eq.12) then
        conversion = 1./1000.      !!Convert millimeters to meters
      else if (unit_in.eq.14 .and. unit_out.eq.13) then
        conversion = 1./10.        !!Convert millimeters to centimeters
      else if (unit_in.eq.14 .and. unit_out.eq.15) then
        conversion = 1./1000000.   !!Convert millimeters to kilometers
      else if (unit_in.eq.15 .and. unit_out.eq.11) then
        conversion = 0.00328084    !!Convert kilometers to feet
      else if (unit_in.eq.15 .and. unit_out.eq.12) then
        conversion = 1000.        !!Convert kilometers to meters
      else if (unit_in.eq.15 .and. unit_out.eq.13) then
        conversion = 100000.      !!Convert kilometers to centimeters
      else if (unit_in.eq.15 .and. unit_out.eq.14) then
        conversion = 1000000.     !!Convert kilometers to milimeters



!     Determine the weight unit conversion
      else if (unit_in.eq.21 .and. unit_out.eq.22) then
        conversion = 1000.        !!Convert kilograms to grams
      else if (unit_in.eq.21 .and. unit_out.eq.23) then
        conversion = 1000000.     !!Convert kilograms to milligrams
      else if (unit_in.eq.21 .and. unit_out.eq.24) then
        conversion = 1000000000.  !!Convert kilograms to micrograms

      else if (unit_in.eq.22 .and. unit_out.eq.21) then
        conversion = 1./1000.     !!Convert grams to kilograms
      else if (unit_in.eq.22 .and. unit_out.eq.23) then
        conversion = 1000.        !!Convert grams to milligrams
      else if (unit_in.eq.22 .and. unit_out.eq.24) then
        conversion = 1000000.     !!Convert grams to micrograms

      else if (unit_in.eq.23 .and. unit_out.eq.21) then
        conversion = 1./1000000.  !!Convert milligrams to kilograms
      else if (unit_in.eq.23 .and. unit_out.eq.22) then
        conversion = 1./1000.     !!Convert milligrams to grams
      else if (unit_in.eq.23 .and. unit_out.eq.24) then
        conversion = 1000.        !!Convert milligrams to micrograms

      else if (unit_in.eq.24 .and. unit_out.eq.21) then
        conversion = 1./1000000000.!!Convert micrograms to kilograms
      else if (unit_in.eq.24 .and. unit_out.eq.22) then
        conversion = 1./1000000.  !!Convert micrograms to grams
      else if (unit_in.eq.24 .and. unit_out.eq.23) then
        conversion = 1./1000.     !!Convert micrograms to milligrams



!     Determine the area unit conversion
      else if (unit_in.eq.31 .and. unit_out.eq.32) then
        conversion = 1./10.7639   !!Convert square feet to square meters
      else if (unit_in.eq.31 .and. unit_out.eq.33) then
```

```
        conversion = 1./107639.    !!Convert square feet to hectacres

      else if (unit_in.eq.32 .and. unit_out.eq.31) then
        conversion = 10.7639       !!Convert square meters to square feet
      else if (unit_in.eq.32 .and. unit_out.eq.33) then
        conversion = 0.0001        !!Convert square meters to hectacres

      else if (unit_in.eq.33 .and. unit_out.eq.31) then
        conversion = 107639.       !!Convert hectacres to square feet
      else if (unit_in.eq.33 .and. unit_out.eq.32) then
        conversion = 10000.        !!Convert hectacres to square meters


      else
        conversion = 1.            !!Conversion Error, conversion not accounted for
      endif


!     Convert line by line the array into its new units
      do i=1,asize
        array(i) = array(i) * (conversion**magnitude)
      enddo

      return
      end
```

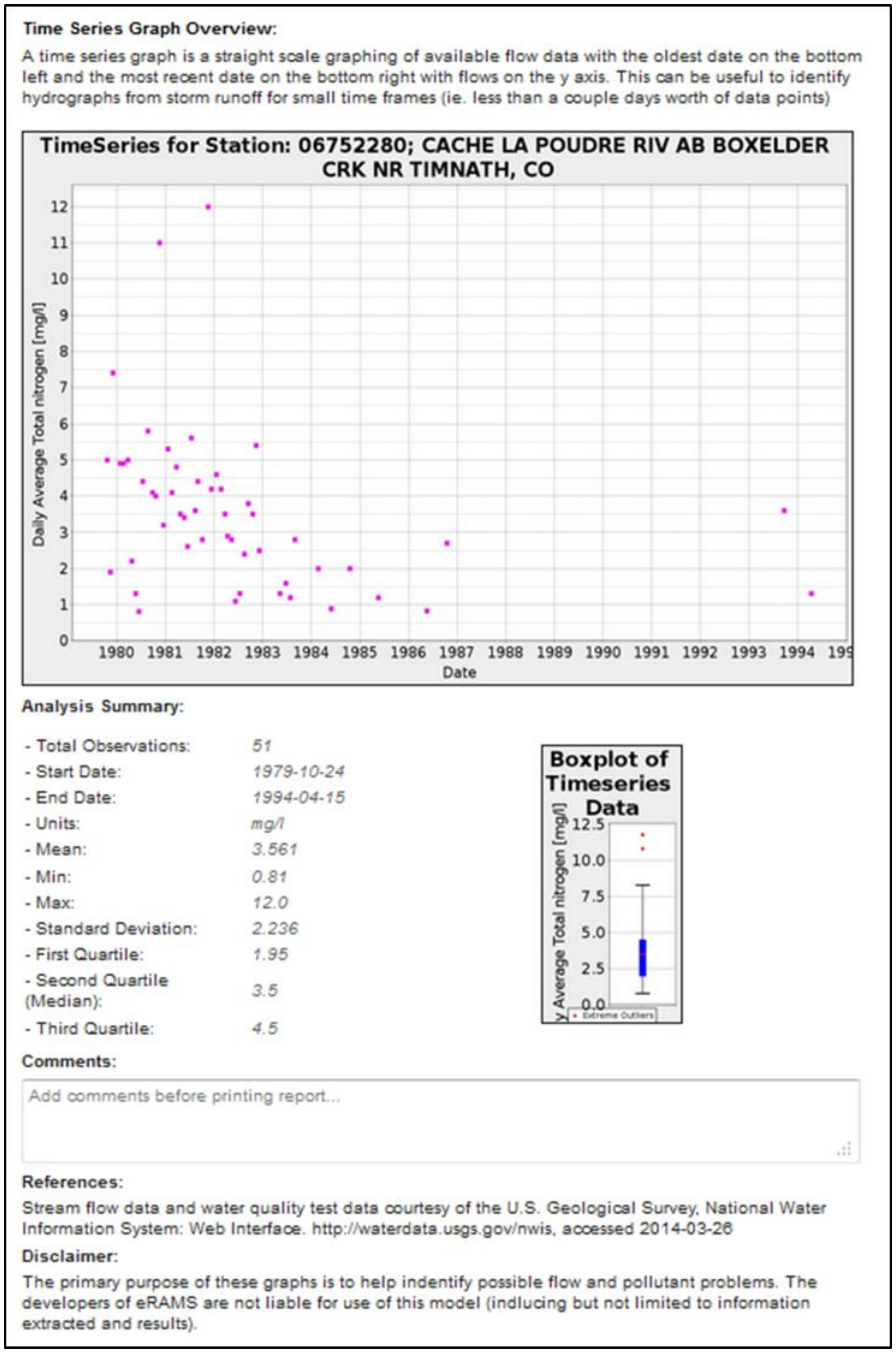
APPENDIX V: COMPREHENSIVE FLOW ANALYSIS EXAMPLE OUTPUT



Station ID:	06752280
Station Name:	CACHE LA POUDRE RIV AB BOXELDER CRK NR TIMNATH, CO
Supervising Agency:	USGS
Latitude:	40.551927
Longitude:	-105.011365
State:	COLORADO
County:	Larimer County
Hydrologic Unit Code:	10190007
Drainage Area:	1244 sq. mi
Elevation:	4860 ft
Elevation Datum:	NGVD29

To upload and analyze your own data click here.

Figure 47: eRAMS Graphical User Interface (GUI) to CFA Tool


TimeSeries for Station: 06752280; CACHE LA POUDRE RIV AB BOXELDER CRK NR TIMNATH, CO
Daily Average Total nitrogen [mg/l]
Date
Boxplot of Timeseries Data
y Average Total nitrogen [mg/l]
Extreme Outliers

Time Series Graph Overview:

A time series graph is a straight scale graphing of available flow data with the oldest date on the bottom left and the most recent date on the bottom right with flows on the y axis. This can be useful to identify hydrographs from storm runoff for small time frames (ie. less than a couple days worth of data points)

Analysis Summary:

- Total Observations:	*51*
- Start Date:	*1979-10-24*
- End Date:	*1994-04-15*
- Units:	*mg/l*
- Mean:	*3.561*
- Min:	*0.81*
- Max:	*12.0*
- Standard Deviation:	*2.236*
- First Quartile:	*1.95*
- Second Quartile (Median):	*3.5*
- Third Quartile:	*4.5*

Comments:

Add comments before printing report...

References:

Stream flow data and water quality test data courtesy of the U.S. Geological Survey, National Water Information System: Web Interface. http://waterdata.usgs.gov/nwis, accessed 2014-03-26

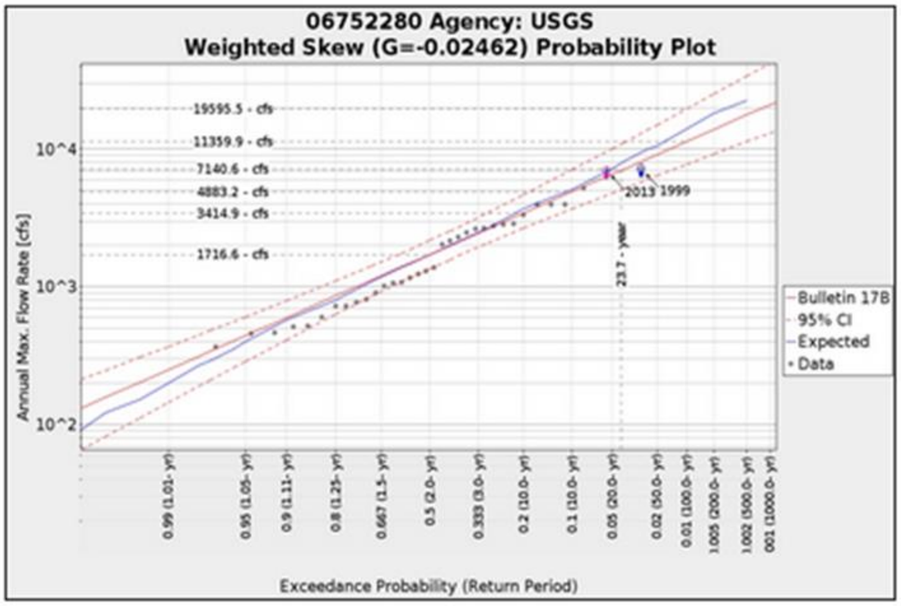
Disclaimer:

The primary purpose of these graphs is to help indentify possible flow and pollutant problems. The developers of eRAMS are not liable for use of this model (indlucing but not limited to information extracted and results).

Figure 48: CFA Example Report Output for Time series and Statistics Analysis

Flood Analysis Results:

The flood analysis follows the USGS Bulletin 17B (IACWD 1982) methodology for fitting a Log-Pearson Type III distribution to available annual flood data. The graph contains the Bulletin 17B fitted data and its corresponding 95% confidence interval, as well as the historic annual flood values. The flood analysis then estimates flood flow value (cfs) for standard return periods, which are summarized in the table below.



Return Period	Expected	Lower 95% CI for B17	B17	Upper 95% CI for B17
(year)	(cfs)	(cfs)	(cfs)	(cfs)
200	18216.4	9197.5	13882.6	25061.2
100	14069.2	7742.5	11359.9	19595.5
50	10544.5	6404.5	9120.3	14985.2
40	9895.2	5997.2	8455.7	13667.0
25	8177.4	5174.3	7140.6	11134.4
20	7201.3	4799.5	6554.8	10041.4
10	5100.7	3694.5	4883.2	7060.0
5.0	3660.3	2664.5	3414.9	4643.1
2.0	1716.6	1355.6	1716.6	2174.3
1.5	1189.6	929.4	1208.5	1526.2
1.25	799.6	631.8	858.8	1100.5

Analysis Summary:

- Total Observations:	34
- Start:	1980
- End:	2013
- Regional Skewness:	-0.1104

Comments:

Add comments before printing report...

References:

Stream flow data and water quality test data courtesy of the U.S. Geological Survey, National Water Information System: Web Interface. http://waterdata.usgs.gov/nwis, accessed 03/26/2014

Interagency Advisory Committee on Water Data (IACWD). 1982. "Guidelines for determining flood flow frequency." *Bulletin No. 17B* (revised and corrected), Hydrology Subcommittee, Washington, D.C.

Water Resources Council, Hydrology Committee. 1967. "A Uniform Technique for Determining Flood Flow Frequencies." *Bulletin No. 15*, Washington, D.C.

Disclaimer:

The primary purpose of these graphs is to help indentify possible flow and pollutant problems. The developers of eRAMS are not liable for use of this model (inlducing but not limited to information extracted and results).

Figure 49: CFA Example Report Output for Flood Analysis

The drought analysis begins by calculating annual flow values from available average daily flow data. Then a drought limit is calculated as the long-term average of annual flow data. Figure 1 contains an annual time series of the flow data along with the calculated drought limit as a reference.

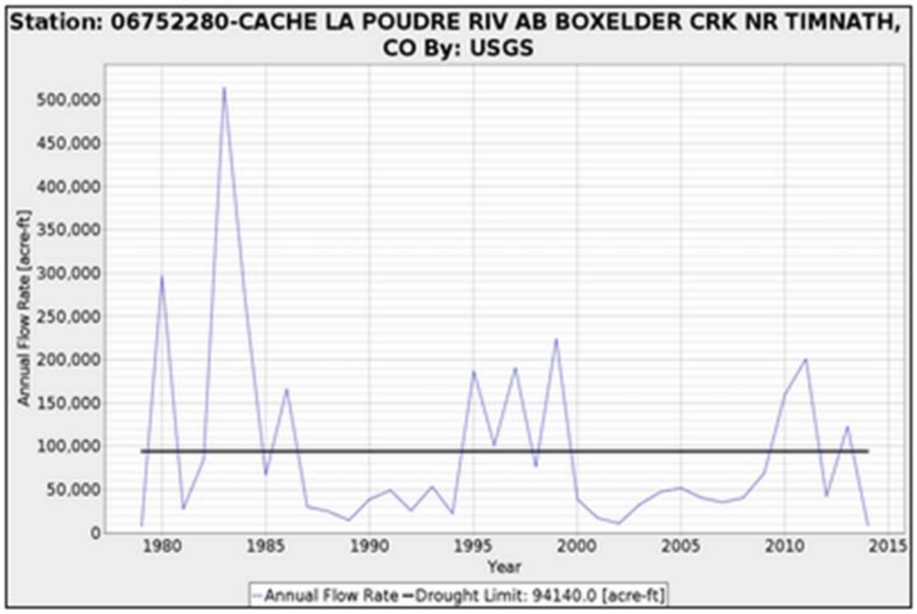


Figure 1: Annual Flow Rate and Drought Limit

Figure 2 contains a second time series containing the annual surplus or deficit between the supplied annual flow and the drought demand limit; this is meant to highlight the occurrence of droughts.

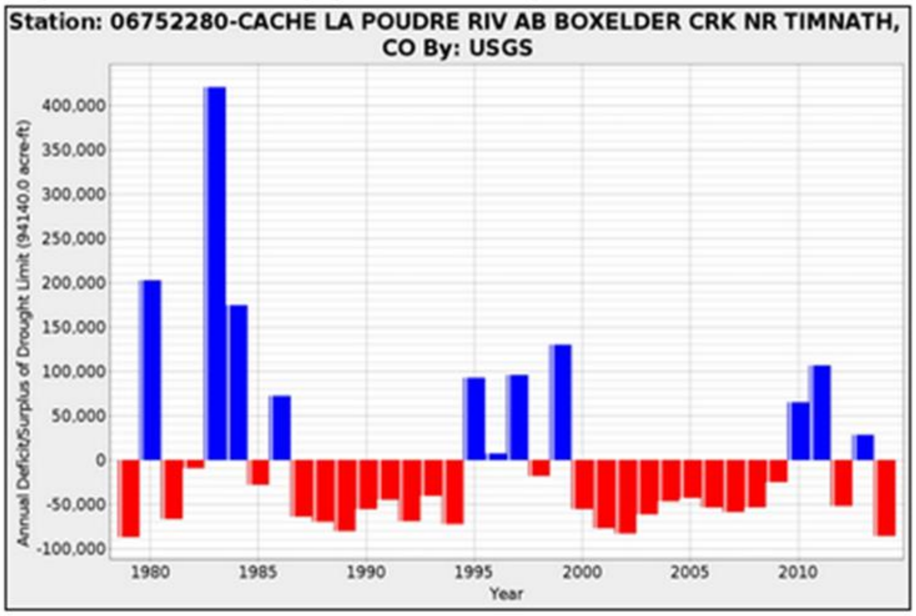


Figure 2: Annual Flow Deficit/Surplus

Then the annual flow data is converted to only its stochastic component (stochastic data = (annual data - mean)/standard deviation). Subsequently, a Box-Cox transformation converts the stochastic data into a normally distributed dataset. Thereafter, an Auto-Regressive, AR(p), model is fitted to the dataset. This allows a forecasting of the minimal observed data to a larger sample size, which in turn allows for a statistical analysis of the droughts. Equation 1 below is the form of the fitted AR(p) model.

Figure 50: CFA Example Report Output for Drought Analysis, Part 1

Equation 1: AR(p) Model

$$\hat{y}_t = \sum_{i=1}^{p} \varphi_i \hat{y}_{t-i} + error_t$$

Figure 3 contains a plot of the original annual data verses the predicted model data to illustrate the correlation between the datasets. If the correlation is poor then further modifications need to be made to the regression model in order to improve the reliability of the drought analysis.

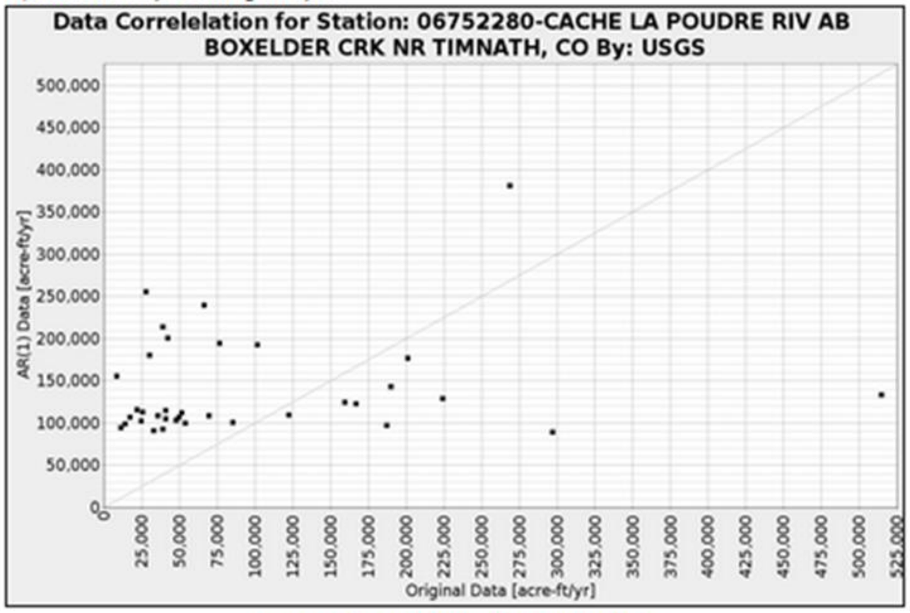


Figure 3: Fit of AR(p) Model to Original Dataset

Figure 4 contains a plot of the original data and the first portion of the 100,000 year projected dateset used to analyze the drought impacts. This projected dataset is large to allow sufficient 'droughts' to occur illustrating high recurrence interval droughts that cannot be calculated from minimal observed data. The first 100 years of this dataset are not used in the analysis and a dropped as a model warm-up period. This allows for the model to operate independent of initial conditions.

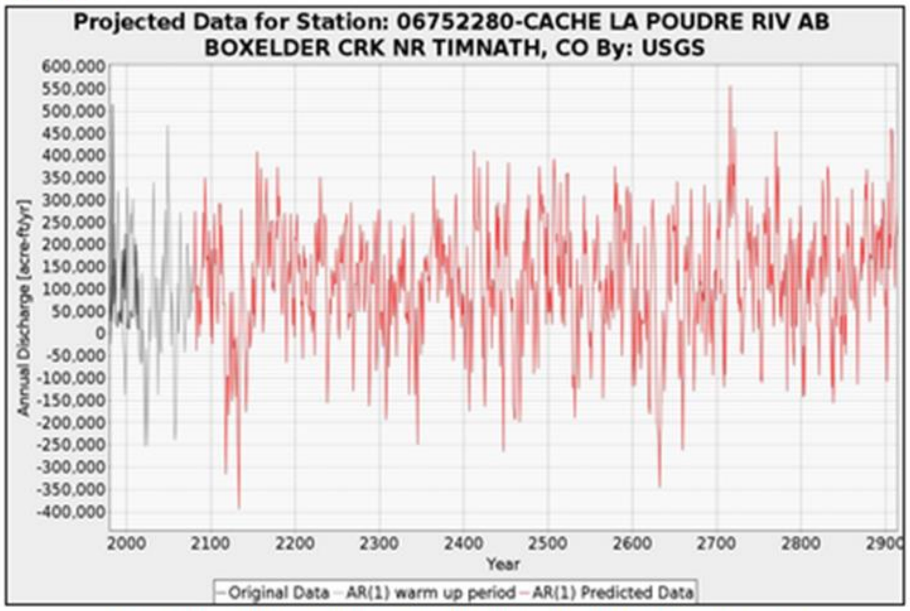


Figure 4: AR(p) Model Prediction with Original Dataset and Model Warm-Up Period

Figure 51: CFA Example Report Output for Drought Analysis, Part 2

Next the drought analysis uses the projected dateset to calculate the average recurrence interval of the 1yr, 2yr, 3yr, etc. droughts. These droughts are then categorized by their amount of drought deficit (supplied annual flow - drought demand limit) and illustrated in Figure 5. The original data and its corresponding recurrence intervals are included in Figure 5 as well to illustrate the fit of the predicted data to that of the observed data. If the fit is poor, a better correlation of the regression model will likely improve the fit of the drought recurrence intervals.

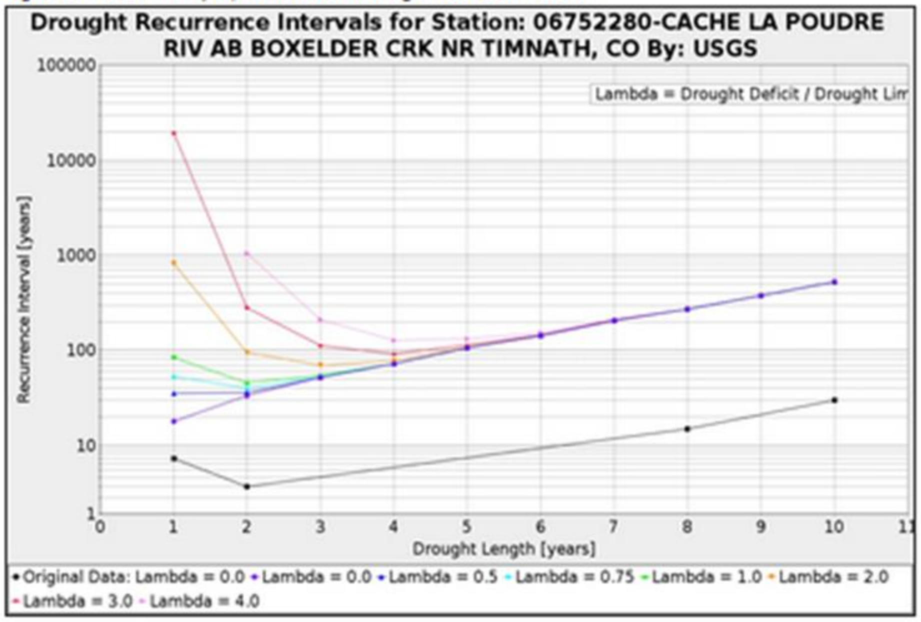


Figure 5: Predicted Drought Recurrence Interval, Length, and Deficit-(relative to the drought limit)

Analysis Summary:

- Total Observations:	*12542*
- Start:	*1979-10-01*
- End:	*2014-03-25*

Comments:

Add comments before printing report...

References:

Stream flow data and water quality test data courtesy of the U.S. Geological Survey, National Water Information System: Web Interface. http://waterdata.usgs.gov/nwis, accessed 03/26/2014

Salas, Jose D., Chongjin Fu, Antonino Cancelliere, Dony Dustin, Dennis Bode, Andy Pineda, and Esther Vincent. 2005. "Characterizing the Severity and Risk of Drought in the Poudre River, Colorado." *Journal of Water Resources Planning and Management* 131(5): 383-393.

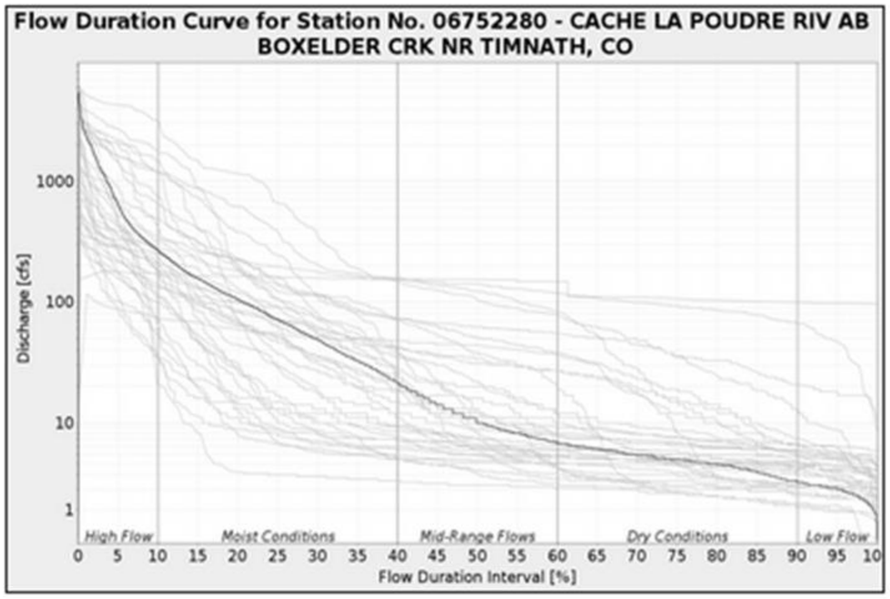
Salas, Jose D. 1993. "Chapter 19: Analysis and Modeling of Hydrologic Time Series." *The McGraw Hill handbook of hydrology.* D. R. Maidment, ed., McGraw-Hill New York

The primary purpose of these graphs is to help indentify possible flow and pollutant problems. The developers of eRAMS are not liable for use of this model (inclucing but not limited to information extracted and results).

Figure 52: CFA Example Report Output for Drought Analysis, Part 3

Flow Duration Curve Overview:

A flow duration curve (FDC) is the ranked graphing of river flows on a scale of percent exceedence. For example a flow value associated with the flow interval of 15% means that particular flow value is met or exceeded only 15% of the time. This graph is meant to give a quick overview of the flow ranges, variability, and probability of flows of a river segment during the different flow periods of a river; which are High Flows from 0 to 10 percent flow interval, Moist Conditions 10-40, Mid-Range Flows 40-60, Dry Conditions 60-90, and Low Flows 90-100 (Cleland 2003). The grey graphed lines are duration curves for each individual year within the analysis period.



Analysis Summary:

- Discharge Observations:	*12542*
- Start Date:	*1970-10-01*
- End Date:	*2014-03-25*

Comments:

Add comments before printing report...

References:

Stream flow data and water quality test data courtesy of the U.S. Geological Survey, National Water Information System: Web Interface. http://waterdata.usgs.gov/nwis, accessed 2014-03-26

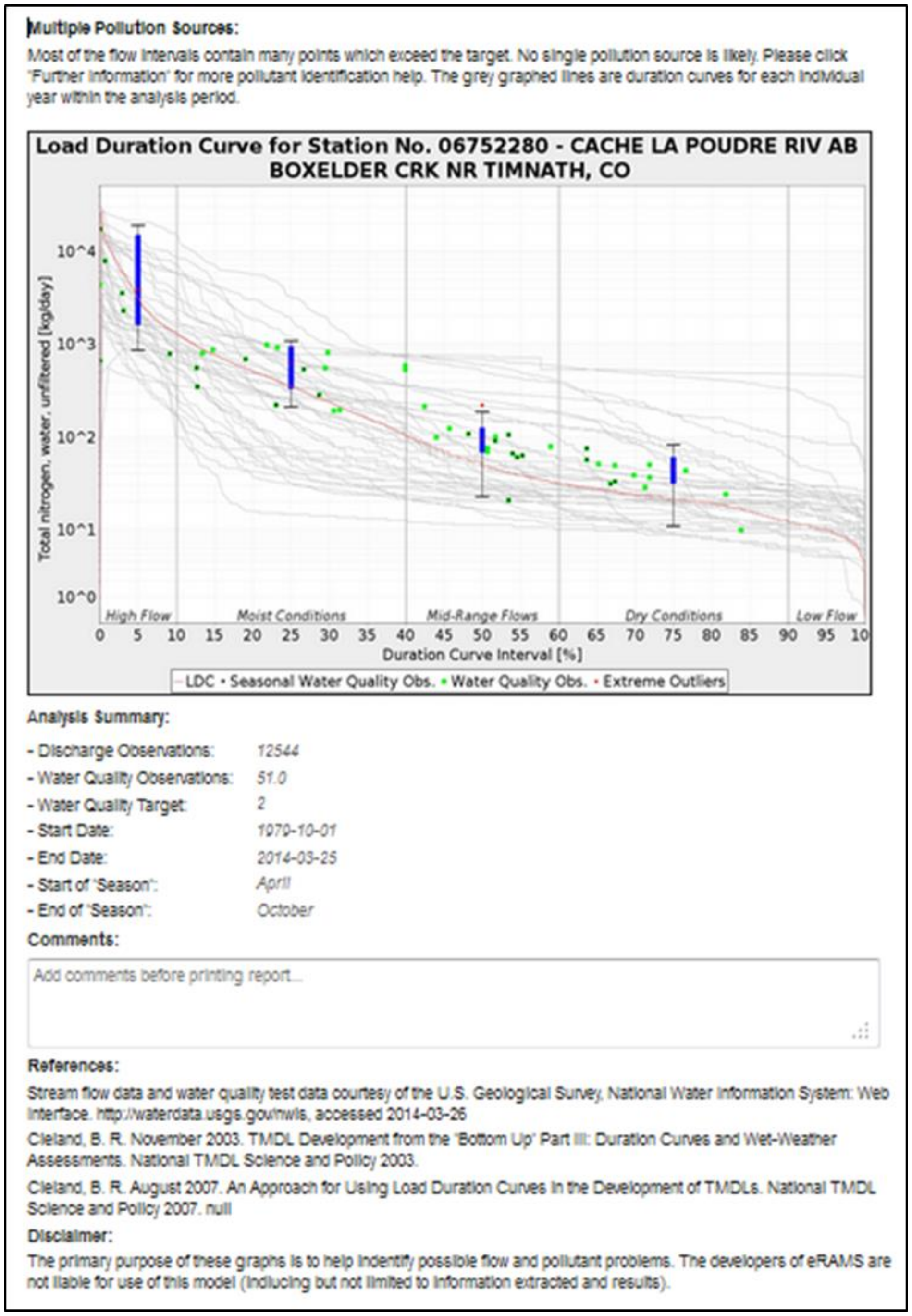
Cleland, B. R. November 2003. TMDL Development from the "Bottom Up" Part III: Duration Curves and Wet-Weather Assessments. National TMDL Science and Policy 2003.

Cleland, B. R. August 2007. An Approach for Using Load Duration Curves in the Development of TMDLs. National TMDL Science and Policy 2007. null

Disclaimer:

The primary purpose of these graphs is to help indentify possible flow and pollutant problems. The developers of eRAMS are not liable for use of this model (indlucing but not limited to information extracted and results).

Figure 53: CFA Example Report Output for Flow Duration Curve Analysis


Load Duration Curve for Station No. 06752280 - CACHE LA POUDRE RIV AB BOXELDER CRK NR TIMNATH, CO
Total nitrogen, water, unfiltered [kg/day]
10^0
10^1
10^2
10^3
10^4
High Flow
Moist Conditions
Mid-Range Flows
Dry Conditions
Low Flow
0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 10
Duration Curve Interval [%]
LDC • Seasonal Water Quality Obs. • Water Quality Obs. • Extreme Outliers

Multiple Pollution Sources:

Most of the flow intervals contain many points which exceed the target. No single pollution source is likely. Please click "Further Information" for more pollutant identification help. The grey graphed lines are duration curves for each individual year within the analysis period.

Analysis Summary:

- Discharge Observations:	*12544*
- Water Quality Observations:	*51.0*
- Water Quality Target:	*2*
- Start Date:	*1970-10-01*
- End Date:	*2014-03-25*
- Start of "Season":	*April*
- End of "Season":	*October*

Comments:

Add comments before printing report...

References:

Stream flow data and water quality test data courtesy of the U.S. Geological Survey, National Water Information System: Web Interface. http://waterdata.usgs.gov/nwis, accessed 2014-03-26

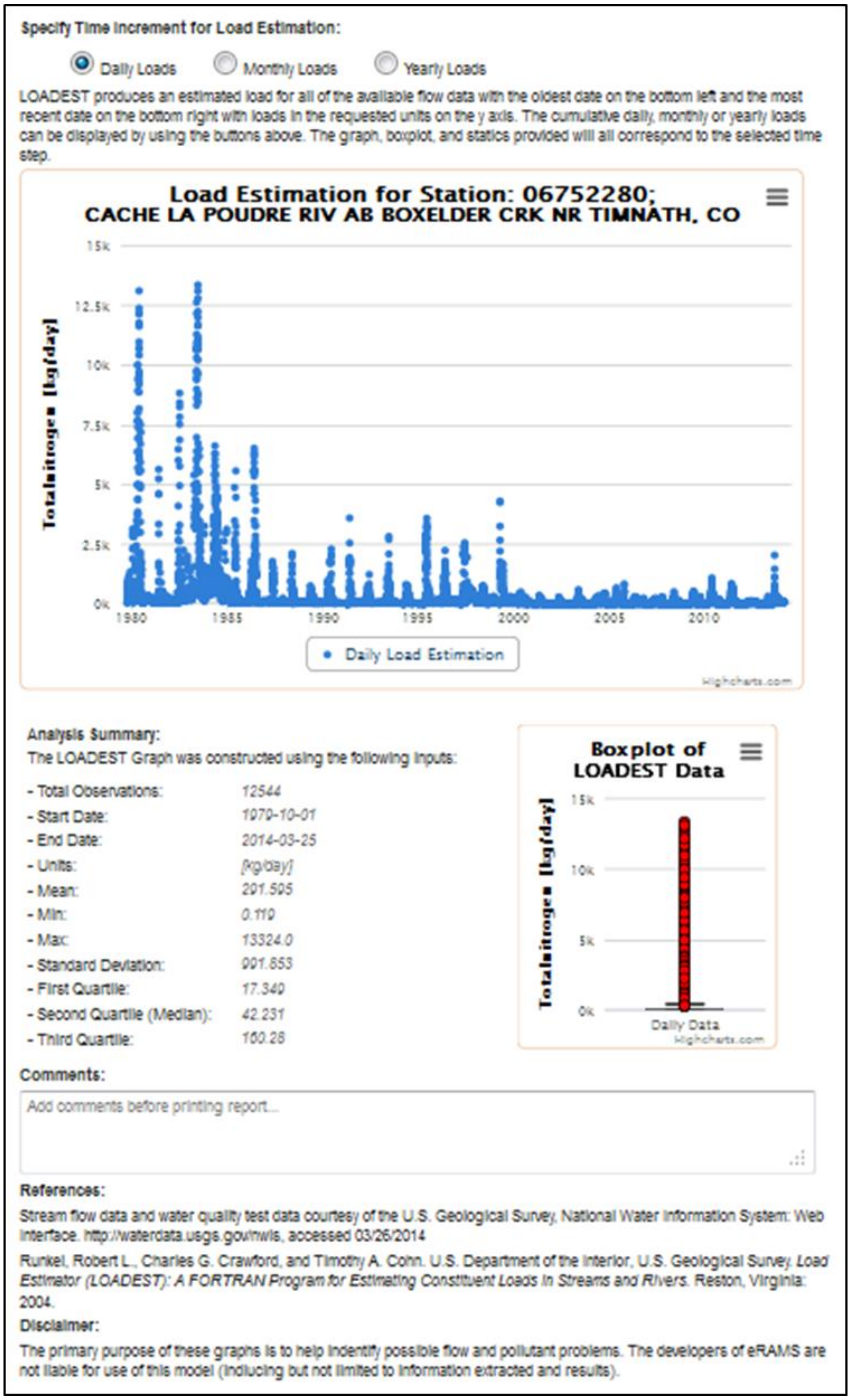
Cleland, B. R. November 2003. TMDL Development from the 'Bottom Up' Part III: Duration Curves and Wet-Weather Assessments. National TMDL Science and Policy 2003.

Cleland, B. R. August 2007. An Approach for Using Load Duration Curves in the Development of TMDLs. National TMDL Science and Policy 2007. null

Disclaimer:

The primary purpose of these graphs is to help indentify possible flow and pollutant problems. The developers of eRAMS are not liable for use of this model (inlducing but not limited to information extracted and results).

Figure 54: CFA Example Report Output for Load Duration Curve Analysis



Specify Time Increment for Load Estimation:

[✓ Daily Loads] Monthly Loads Yearly Loads

LOADEST produces an estimated load for all of the available flow data with the oldest date on the bottom left and the most recent date on the bottom right with loads in the requested units on the y axis. The cumulative daily, monthly or yearly loads can be displayed by using the buttons above. The graph, boxplot, and statics provided will all correspond to the selected time step.

Analysis Summary:

The LOADEST Graph was constructed using the following inputs:

- Total Observations:	*12544*
- Start Date:	*1970-10-01*
- End Date:	*2014-03-25*
- Units:	*[kg/day]*
- Mean:	*201.595*
- Min:	*0.110*
- Max:	*13324.0*
- Standard Deviation:	*901.853*
- First Quartile:	*17.340*
- Second Quartile (Median):	*42.231*
- Third Quartile:	*150.28*

Comments:

Add comments before printing report...

References:

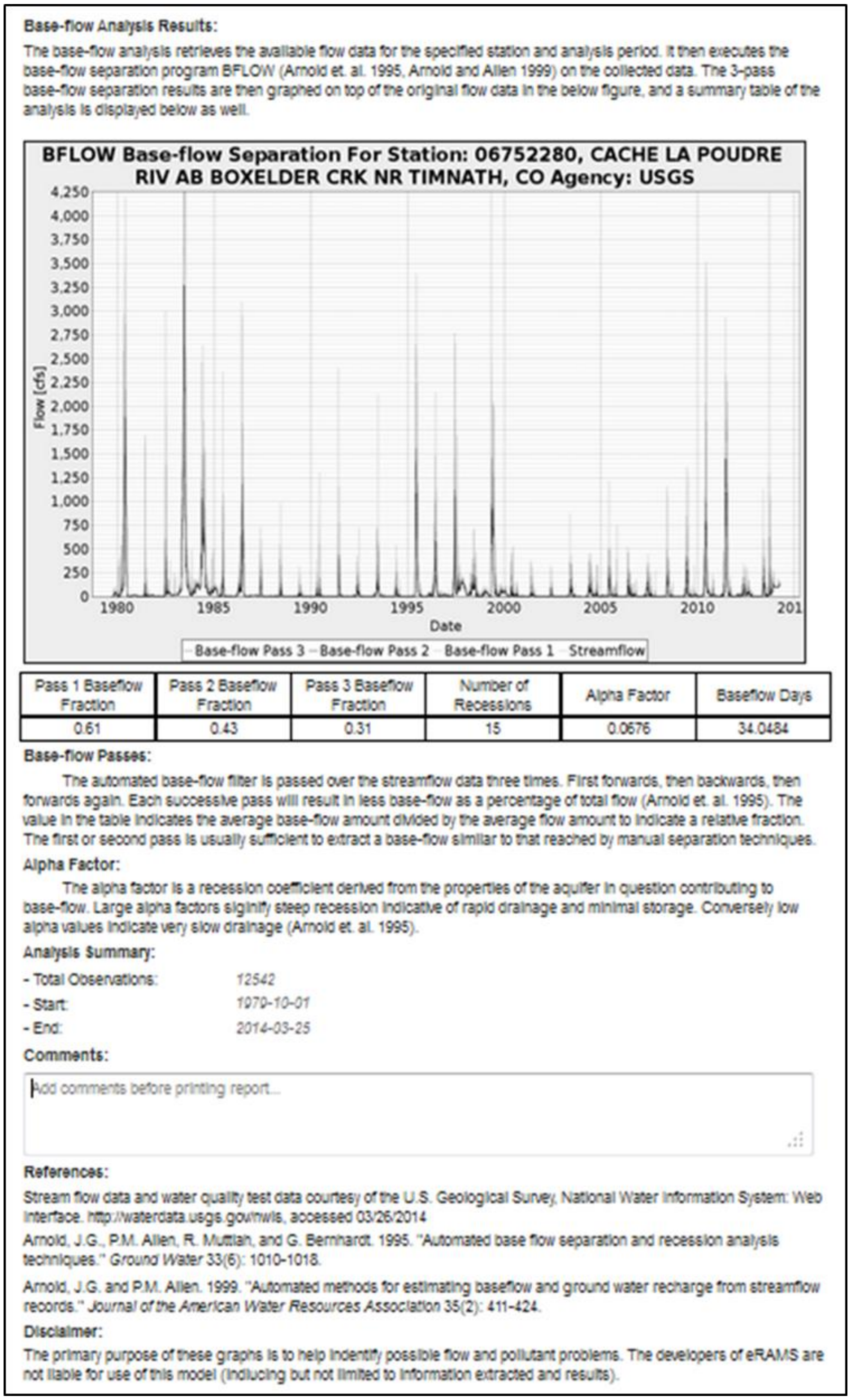
Stream flow data and water quality test data courtesy of the U.S. Geological Survey, National Water Information System: Web Interface. http://waterdata.usgs.gov/nwis, accessed 03/26/2014

Runkel, Robert L., Charles G. Crawford, and Timothy A. Cohn. U.S. Department of the Interior, U.S. Geological Survey. *Load Estimator (LOADEST): A FORTRAN Program for Estimating Constituent Loads in Streams and Rivers.* Reston, Virginia: 2004.

Disclaimer:

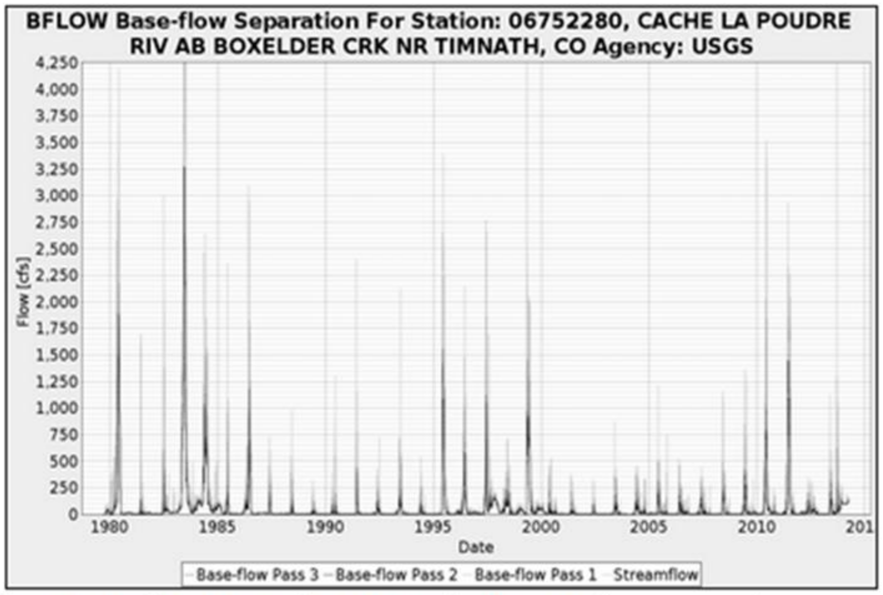
The primary purpose of these graphs is to help indentify possible flow and pollutant problems. The developers of eRAMS are not liable for use of this model (inlducing but not limited to information extracted and results).

Figure 55: CFA Example Report Output for Daily Load Estimator (LOADEST) Analysis



Base-flow Analysis Results:

The base-flow analysis retrieves the available flow data for the specified station and analysis period. It then executes the base-flow separation program BFLOW (Arnold et. al. 1995, Arnold and Allen 1999) on the collected data. The 3-pass base-flow separation results are then graphed on top of the original flow data in the below figure, and a summary table of the analysis is displayed below as well.



Pass 1 Baseflow Fraction	Pass 2 Baseflow Fraction	Pass 3 Baseflow Fraction	Number of Recessions	Alpha Factor	Baseflow Days
0.61	0.43	0.31	15	0.0676	34.0484

Base-flow Passes:

The automated base-flow filter is passed over the streamflow data three times. First forwards, then backwards, then forwards again. Each successive pass will result in less base-flow as a percentage of total flow (Arnold et. al. 1995). The value in the table indicates the average base-flow amount divided by the average flow amount to indicate a relative fraction. The first or second pass is usually sufficient to extract a base-flow similar to that reached by manual separation techniques.

Alpha Factor:

The alpha factor is a recession coefficient derived from the properties of the aquifer in question contributing to base-flow. Large alpha factors siginify steep recession indicative of rapid drainage and minimal storage. Conversely low alpha values indicate very slow drainage (Arnold et. al. 1995).

Analysis Summary:

- Total Observations:	*12542*
- Start:	*1979-10-01*
- End:	*2014-03-25*

Comments:

Add comments before printing report...

References:

Stream flow data and water quality test data courtesy of the U.S. Geological Survey, National Water Information System: Web Interface. http://waterdata.usgs.gov/nwis, accessed 03/26/2014

Arnold, J.G., P.M. Allen, R. Muttiah, and G. Bernhardt. 1995. "Automated base flow separation and recession analysis techniques." *Ground Water* 33(6): 1010-1018.

Arnold, J.G. and P.M. Allen. 1999. "Automated methods for estimating baseflow and ground water recharge from streamflow records." *Journal of the American Water Resources Association* 35(2): 411-424.

Disclaimer:

The primary purpose of these graphs is to help indentify possible flow and pollutant problems. The developers of eRAMS are not liable for use of this model (inclucing but not limited to information extracted and results).

Figure 56: CFA Example Report Output for Base-flow Separation (BFLOW) Analysis

APPENDIX VI: COMPREHENSIVE FLOW ANALYSIS (CFA) TECHNICAL MANUAL

1. INTRODUCTION

Stream flow data has become an increasingly important tool for assessing current stream conditions and as a predictor for future conditions. However, there are numerous aspects of stream flow to analyze as well as methods to do so:

- Stream flow variability and availability for water rights and allocations
- Extent and use of flood plains
- Amount of return flows from groundwater to streams, mostly for modeling purposes
- Impact and extent of droughts on municipal water supply

The foundation of all of these analyses is the stream flow record itself, but there is not currently a uniform approach to each of these topics combined into a single comprehensive tool. The manner and implementation of a new tool is of additional importance to its acceptance and usage.

Current stream flow analysis tools and numerical techniques like base-flow separation BFLOW (Arnold et al. 1995; Arnold and Allen 1999), hydrograph separation HYSEP (Sloto and Michele 1996), the Bulletin 17B flood analysis method (IACWD 1982), Web-based Hydrograph Analysis Tool WHAT (Lim et al. 2005), drought analysis (Salas et al. 2005; Mishra and Singh 2011), and others require installation and use of a software package on a single computer or manual data manipulation and calculations. Numerical automation of data analysis can streamline data processing and remove inherent uncertainties in manual data manipulation techniques. Additionally, the benefit of a web-based tool is that it requires no software installation and is platform independent. For these reasons web-based software is much easier to deploy as well as simpler for people to use. A further complication of web-modeling development is the scaling of usage to meet user demands. The utilization of cloud infrastructure allows the intensive calculations to be moved from a single server to one or many cloud-based virtual machines, as needed based on current demand/usage.

With the above features in mind the Comprehensive Flow Analysis, CFA, tool was designed for the Environmental Risk Assessment and Management System, eRAMS. eRAMS is a web-based geospatial analysis tool to facilitate open-source environmental modeling. The web-deployment of eRAMS satisfies the design criteria for no software installation necessary for users. Additionally, eRAMS' utilizes the cloud-based modeling services provided by the Cloud Services Innovation Platform, CSIP (David et al. 2012). The cloud services reached by eRAMS, through CSIP, satisfies the second criteria to utilize virtual machine computation. CSIP provides an open web interface to the models integrated with it, utilizing a Representational State Transfer, REST, web service to facilitate initiating, interacting, and retrieving results from modeling runs.

2. USER MANUAL

- Access
    - The non-login version of CFA is available at www.erams.com/flowanalysis
    - The login version of CFA is available at www.erams.com
        - Log in
        - Go to the Projects section of your profile
        - Start a new project and use the "Flow Analysis" project type
        - Click "GIS/Analysis" to go to the map interface

- Go to the “Flow Analysis” tab of the map
- Search for a flow/water quality monitoring station using either a:
    - Keyword search
    - Point buffer (circular area around a point clicked on the map)
    - Line buffer (circular area around a line drawn on the map)
    - Polygon (an arbitrary shape drawn on the map)
    - Rectangle (a bounding box drawn on the map)
- After finding the station of interest, click on it on the map and select “Flow Analysis Model” on the summary of the station
    - This launches the CFA interface
    - Further instructions are available under “Getting Started”
- Select the type of analysis model you wish to run from the tabs at the top (data, flood, drought, base-flow, duration curves, LOADEST)
    - Provide the request inputs for the model, tips and information are available from the “Help” button
- Click the “Run Model” button
    - If the raw data selected on the interface is desired click “Download Data”
- The results of CFA are then added to a results page in eRAMS and displayed.
    - The result files, graphs, and summary page of model runs are available for download.

3. CURRENT INFRASTRUCTURE

The Environmental Risk Assessment and Management System (eRAMS) website developed by Dr. Mazdak Arabi at Colorado State University was created to facilitate geospatial manipulation of data for environmental modeling. eRAMS works on a web-based geospatial analyst, similar to ArcGIS, to manipulate, model, and share geospatial information. Additionally, multiple models have been linked into eRAMS including watershed delineation, the Soil and Water Analysis Tool (SWAT), a multi-criteria decision analysis tool, data extraction tools, the High Country Solar Platform (HCSP) for determining solar panel feasibility. The CFA tool is accessible on eRAMS through a scalable cloud-based framework called the Cloud Service Infrastructure Platform (CSIP) developed by Olaf David and Wes Lloyd (2013).

The Comprehensive Flow Analysis (CFA) tool was developed by creating and integrating a series of flow analysis methods into a single web tool and interface. CFA includes six flow models: a time series and statistical analysis, a flood analysis, a drought analysis, a base-flow separation tool, a flow and load duration curve tool, and a load estimator tool. The combination of these models into a single program on an open-source-cloud-based platform allows for multiple independent analyses on the same dataset using the same tool without the need to switch programs or re-format input data for a different flow analysis. Beyond simply saving time, CFA creates a standardized approach to the different aspects of flow analysis allowing site to site comparisons of results.

Behind the scenes, a model run of CFA is accomplished by taking the inputs, for example: which model is requested (flood, time series, base-flow, etc.), station ID, begin and end dates, and other information. This is then passed them from eRAMS to CSIP via a representational state transfer, REST using a JavaScript Object Notation (JSON) to list the inputs of the desired CFA run. After receiving the REST request, CSIP initializes a model run of CFA, waits for it to finish executing, then returns the result from CFA back to eRAMS. An outline of this interaction is shown below in Figure 57.

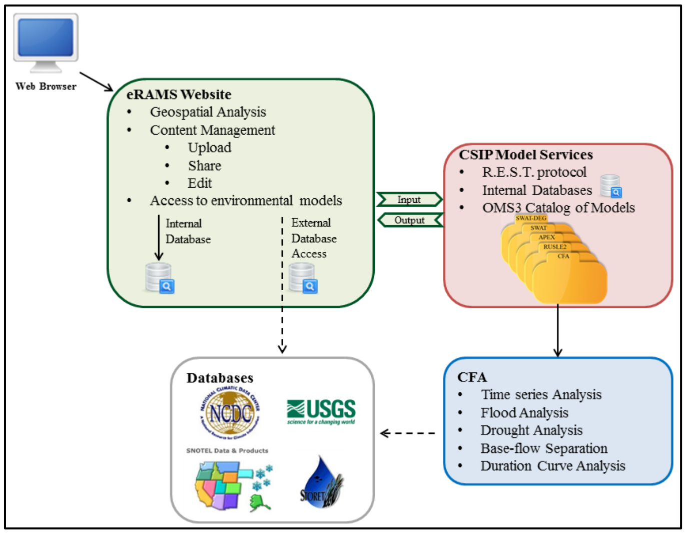


Figure 57: CFA's Interaction with eRAMS, CSIP, and External Databases

4. COMPREHENSIVE FLOW ANALYSIS (CFA) MODELS

Each of the analysis methods included in CFA is summarized below including an explanation of method-specific inputs and outputs.

4.1. TIME SERIES ANALYSIS

The first model included in CFA is a simple time series analysis. The Time Series Analysis Tool graphs temporal changes in available flow or water quality data for any given station within the specified time period of interest. Time series also provides a summary of the statistics of the graphed data, including its min, max, median, mean, upper and lower quartiles, and standard deviation. An example of the output from the Time Series Analysis Tool is shown below in Figure 58.

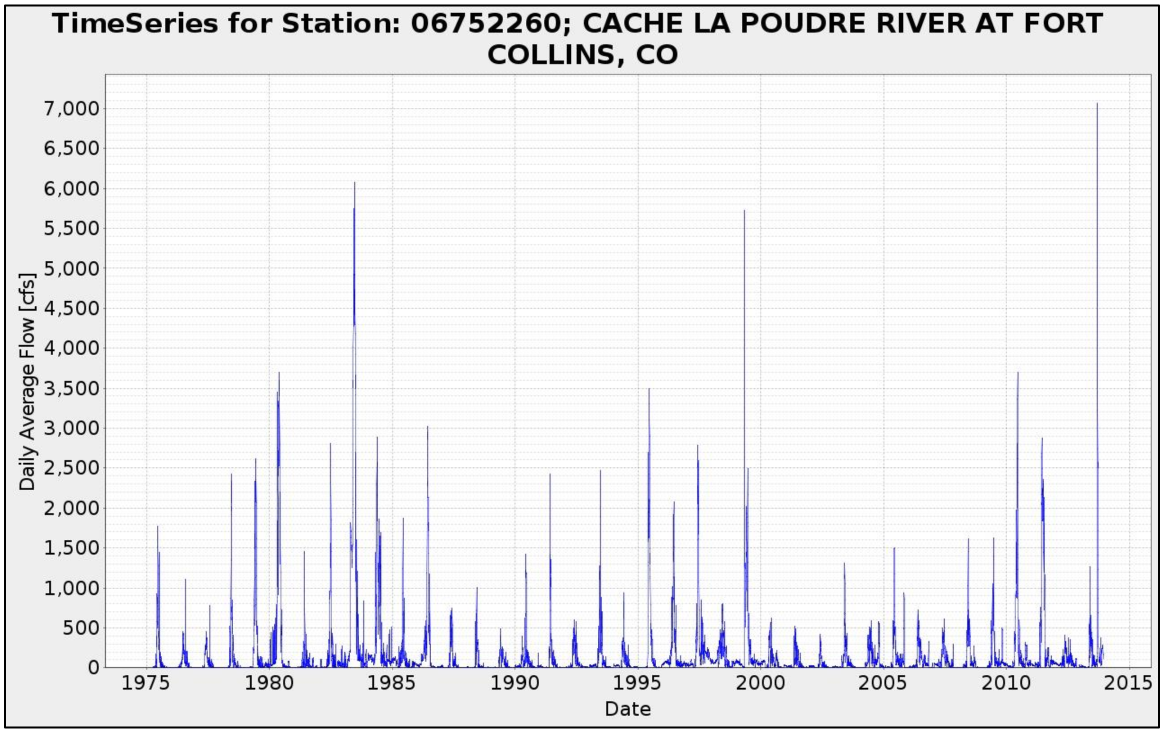


Figure 58: Example CFA's Time Series Analysis Tool Result Graph

4.2. FLOOD ANALYSIS

Of greater benefit than a simple time series of stream flows, CFA also includes a flood analysis model. The Flood Analysis Tool in CFA follows the USGS Bulletin 17B approach (IACWD 1982) for flood flow frequency analysis of unregulated streams. However, CFA is unable to verify whether the stream gauging stations are unregulated or not. For this reason, as with all models, users should have some prior knowledge about the model and its limitations as well as knowledge of the area of interest. The USGS Bulletin 17B method follows the recommendations of Bulleting 15 (WRC-HC 1967) for flood magnitude/frequency study, in which a Log-Pearson Type III distribution is fitted to available flood data. By fitting a distribution to available data, return periods for unobserved and historic floods can be calculated using the parameters of the fitted distribution. This also allows for flood flows of standard return periods, like the 100-year flood, to be interpolated from the fitted distribution.

Due to the sensitivity of the Log-Pearson Type III distribution to its skewness parameter Bulletin 17B published by the Hydrology Subcommittee of the Interagency Advisory Committee on Water Data (IACWD 1982) recommends the use of both a station skew value, derived from available station data, and a generalized regional skew value. The generalized regional skew value can be found from interpolation of the regional skew map included in the Bulletin 17B documentation (IACWD 1982). For greater accuracy of the regionalized skewness, many states have developed similar maps of their states and surrounding areas based on new regression techniques (see Appendix VII for flood skewness coefficient references). Within CFA the state skewness maps were digitized and interpolated on as well as the Plate I map (IACWD 1982) and combined allowing the regional skewness value for each station to be auto-extracted. The generalized skewness coefficients used in this particular tool are first attempted to be taken from a state agency generated map; then if no state data is available the skewness is take from the Plate I map (IACWD 1982). As per the recommendation of Bulletin 17B (IACWD 1982) the final skewness used in the flood analysis is an average between the station skewness, calculated from the available flood dataset, and the generalized skewness, found as described above. An example of the result of the Flood Analysis Tool can be seen below in Figure 59. Following the figure is an explanation of the methodology in CFA’s flood analysis tool using the Bulletin 17B method.

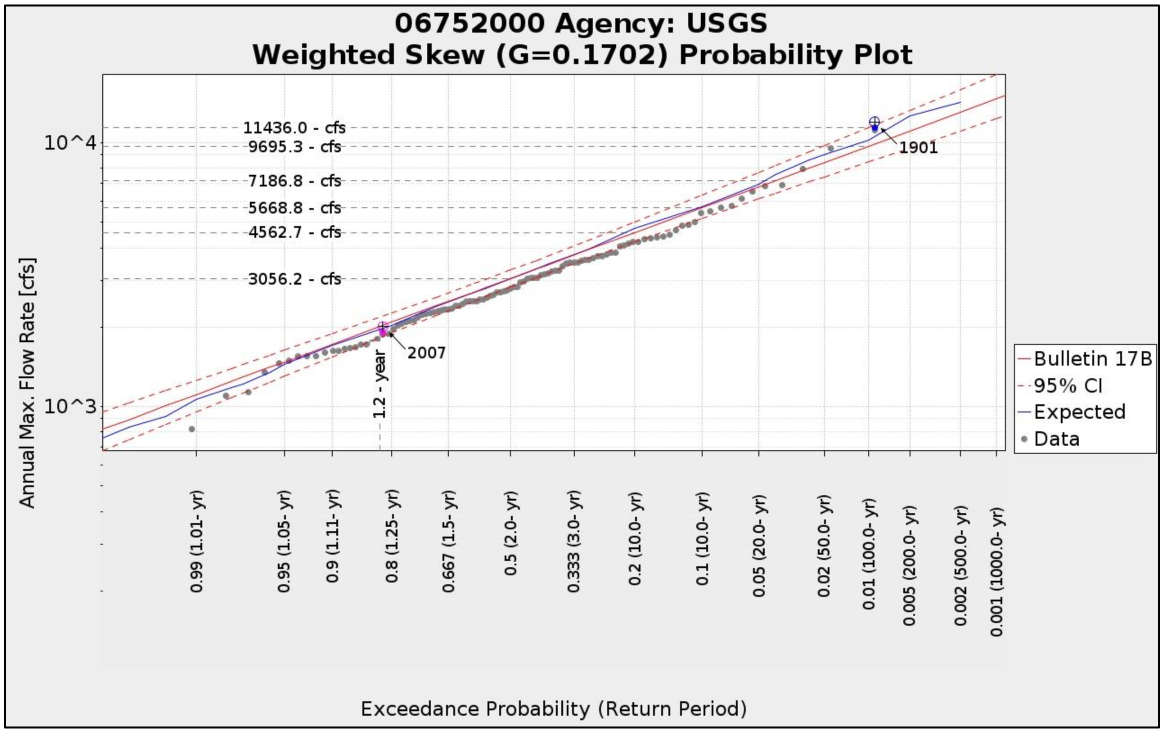


Figure 59: Example CFA's Flood Analysis Tool Result Graph

- The methodology inside CFA's Bulletin 17B method is to first check if there is sufficient data for the analysis (greater than 10 and less than 149 flood peaks).
- Then the statistics (count, Log10 mean, standard deviation, and skewness) for the base dataset are calculated.
- Based on the skewness value the outliers of the dataset (if any) are determined. The statistics of the dataset are then recalculated with the new outlier-removed dataset. If the skewness changes greatly between these steps a warning flag is conveyed to the user.
- Then the frequency/probabilities for each flood are linearly interpolated from the tables provided in the Bulletin 17B documentation (IACWD 1982). If the flood is outside the dataset for interpolation, those values are extrapolated.

- The frequencies/probabilities are then plotted against flood magnitudes in skewed probability space (aka the spacing between the probabilities is not standard unless skewness equals zero).

4.3. DROUGHT ANALYSIS

An opposite, but equally important, aspect of stream flows is the consideration and analysis of droughts from stream flow records. For this reason a generalized drought analysis tool was included in CFA. The drought analysis method included in CFA fits a regression model to historic annual stream flow data and forecasts it to simulate a larger dataset in order to predict high recurrence interval droughts (Salas, et al. 2005). The following is a step by step explanation of the drought analysis method used and example outputs for each step.

The drought analysis begins by calculating annual flow values from available average daily flow data. Figure 60 contains an annual time series of the flow data with the specified annual drought limit as a reference.

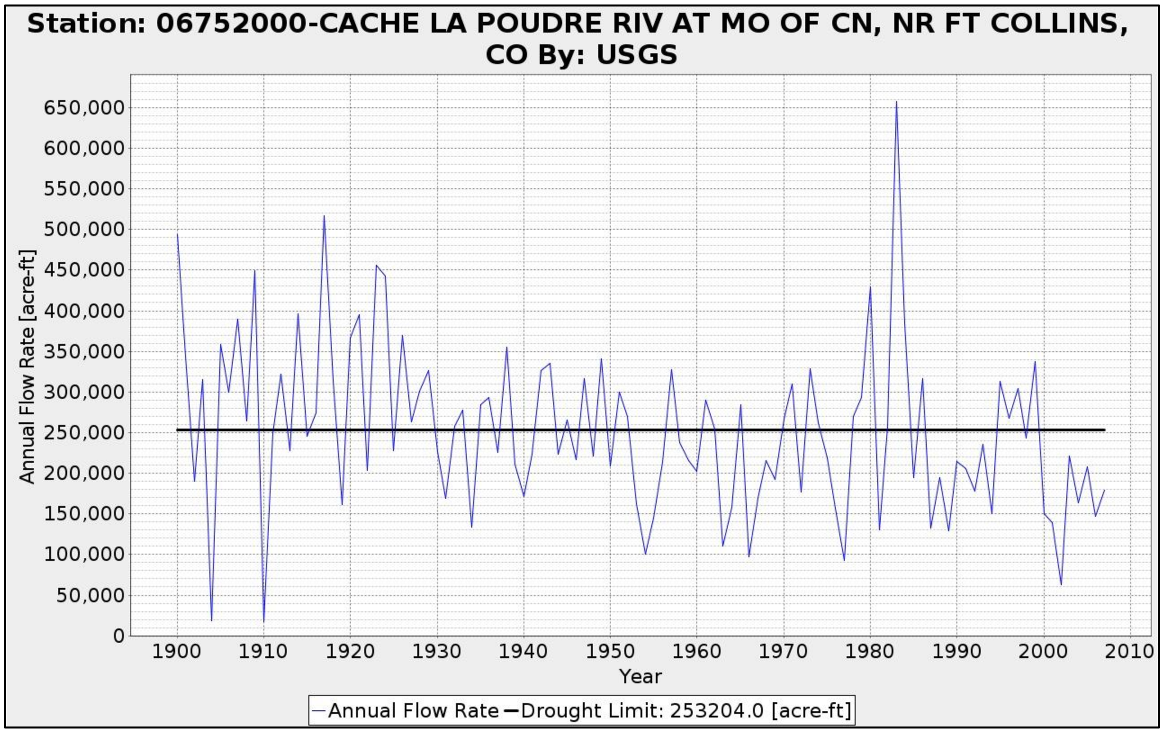


Figure 60: Example CFA's Drought Analysis Tool Result Graph 1

Figure 61 contains a second time series containing the annual surplus or deficit between the supplied annual flow and the drought demand limit; this is meant to highlight the occurrence of droughts.

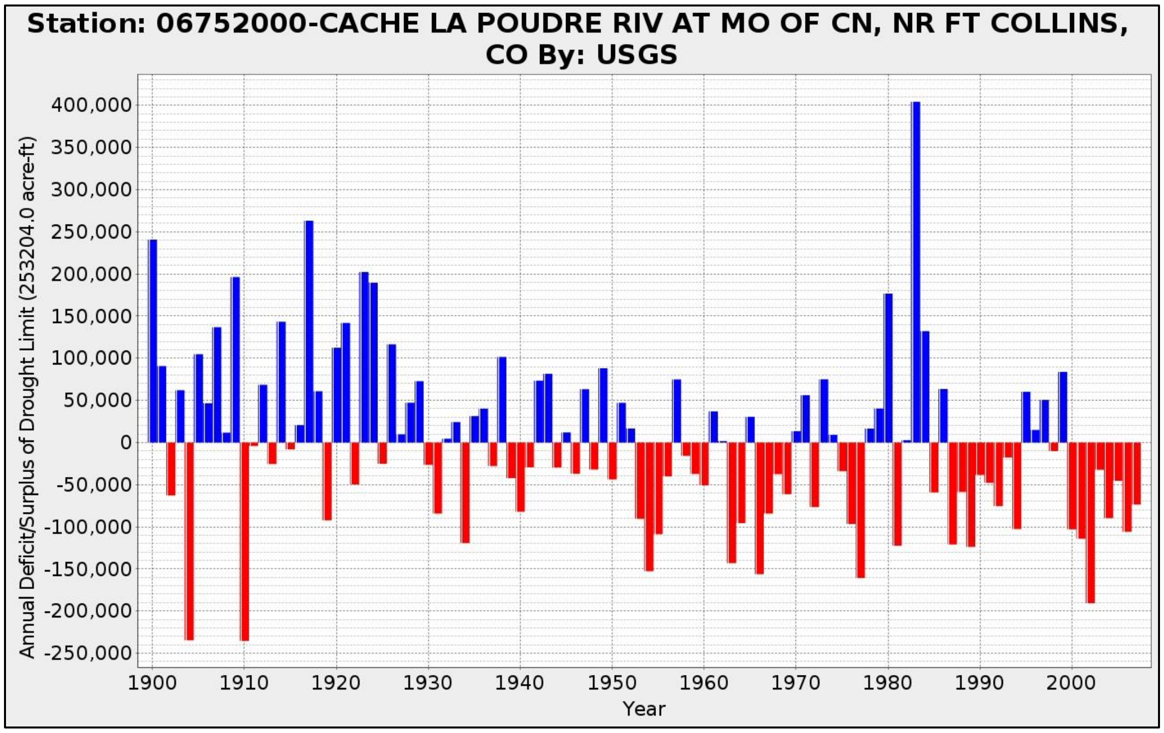


Figure 61: Example CFA's Drought Analysis Tool Result Graph 2

After calculating the annualized flow data, it is then converted to its stochastic component (the mean is subtracted from the data and then divided by the standard deviation). The stochastic data is then transformed into a normalized dataset using a Box-Cox transformation. Then an Auto-Regressive (AR) or Auto-Regressive-Moving-Average (ARMA) model is fitted to the dataset (Salas 1993). The purpose of fitting the regressive model to the stochastic data is to increase the size of the dataset while maintaining its statistical properties, mean and standard deviation. Figure 62 contains a plot of the original annual data versus the predicted model data to illustrate the correlation between the datasets. If the correlation is poor then further modifications need to be made to the regression model in order to improve the reliability of the drought analysis.

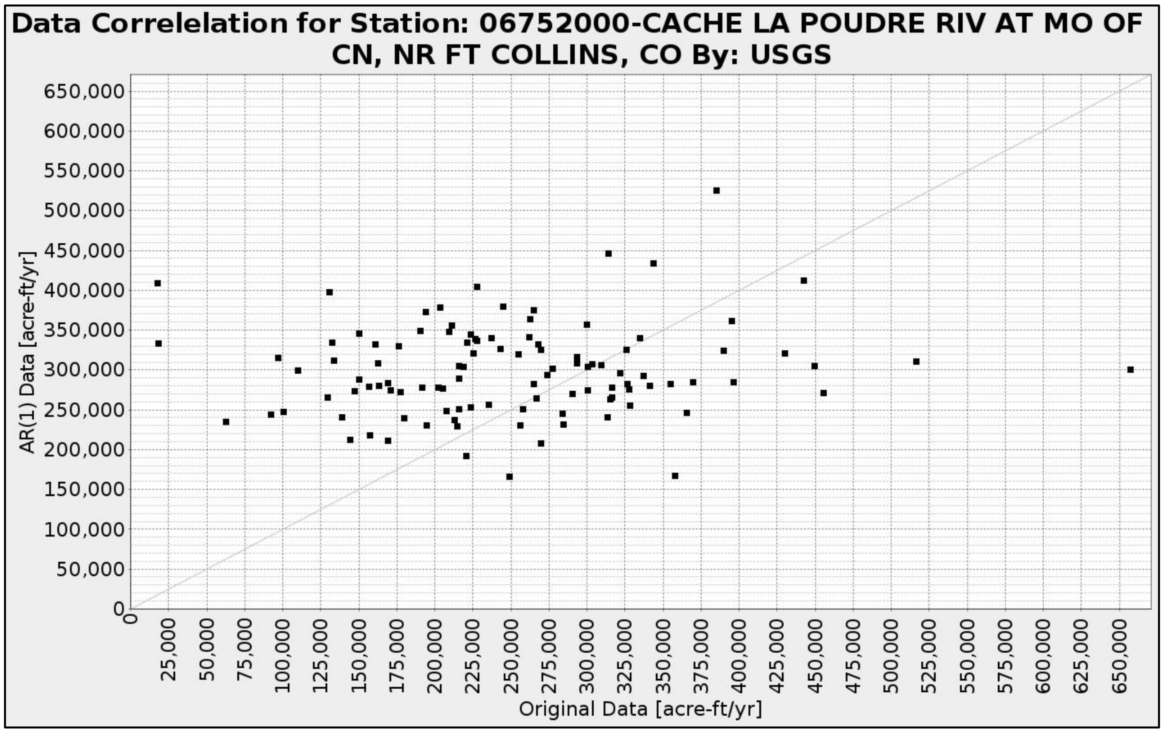


Figure 62: Example CFA's Drought Analysis Tool Result Graph 3

After fitting the regression model, a 100,000 year forecasting is performed using the fitted model to create a dataset sufficiently large to ‘observe’ high recurrence interval droughts. Figure 63 contains a plot of the original data and the first portion of the 100,000 year projected dataset used to analyze the drought impacts. This projected dataset is large to allow sufficient 'droughts' to occur illustrating high recurrence interval droughts that cannot be calculated from minimal observed data. The first 100 years of this dataset are not used in the analysis and a dropped as a model warm-up period. This allows for the model to operate independent of initial conditions.

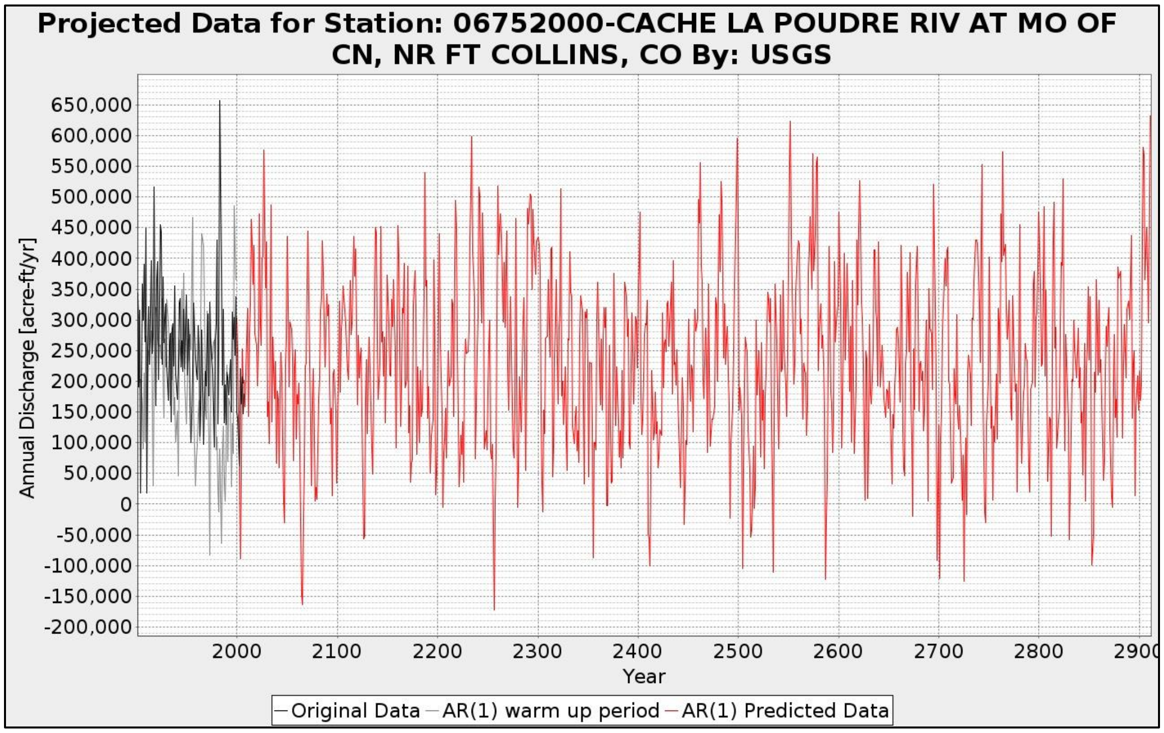


Figure 63: Example CFA's Drought Analysis Tool Result Graph 4

Next the drought analysis uses the projected dataset to calculate the average recurrence interval of the 1yr, 2yr, 3yr, etc. droughts. These droughts are then categorized by their amount of drought deficit (supplied annual flow - drought demand limit) and illustrated in Figure 64. The original data and its corresponding recurrence intervals are included in Figure 64 as well to illustrate the fit of the predicted data to that of the observed data. If the fit is poor, a better correlation of the regression model will likely improve the fit of the drought recurrence intervals.

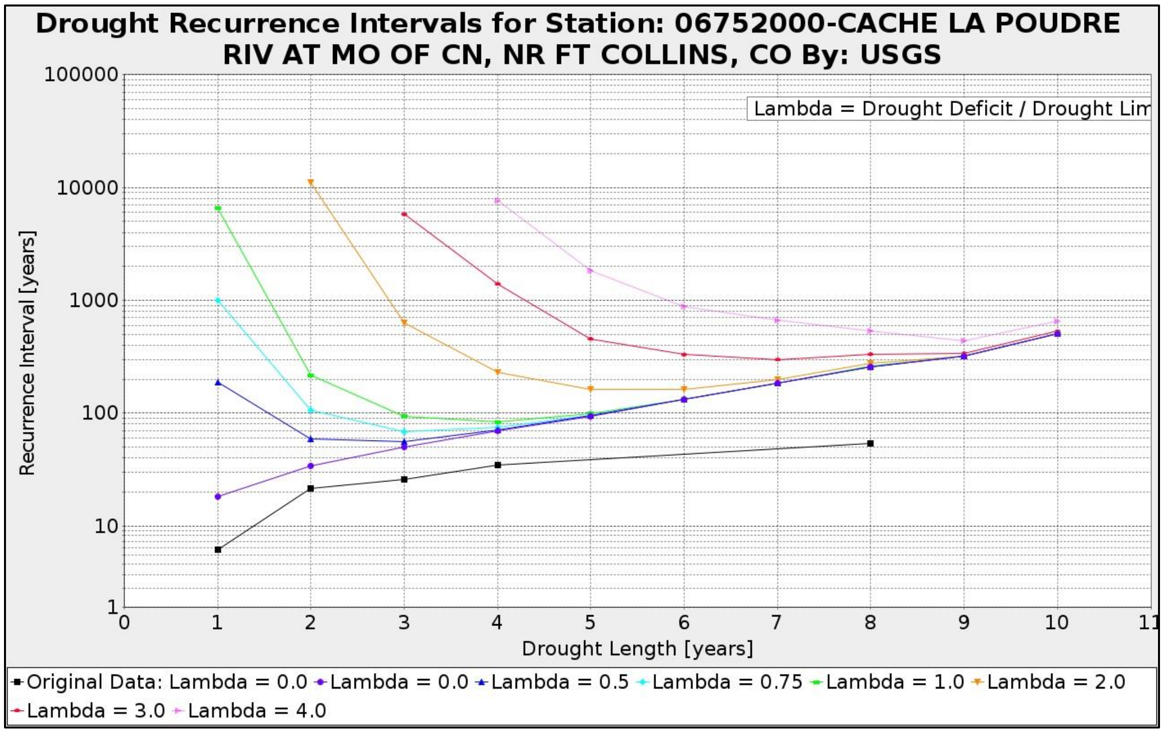


Figure 64: Example CFA's Drought Analysis Tool Result Graph 5

4.4. BASE-FLOW SEPARATION (BFLOW)

Another useful aspect of stream flow analysis and hydrologic modeling of river basins is river base-flow. Rather than write a numerical hydrograph separation tool, CFA has incorporated the numerical base-flow separation program "BFLOW," developed by the by Arnold et al. (1995; Arnold and Allen 1999). BFLOW is an automated digital filter base-flow separation tool which performs a multi-pass separation of base-flow from total stream flow. In order to implement the windows executable BFLOW in CSIP, which uses a Linux platform, the windows emulator WINE (WineHQ 2012) was used within CSIP (Lloyd et al. 2012). For the ease of use, like the rest of the tools in CFA, BFLOW operates on uploaded or auto-extracted data. CFA also automatically generates and formats the data into the necessary input files for the BFLOW executable. Beyond simply performing the analysis and returning the results CFA's base-flow analysis also graphs the outputs of BFLOW's separation overlaid onto total stream flow for a visual understanding of groundwater contributions to stream flow, see Figure 65. The result file of the BFLOW program is available for download like the other flow analysis modes in CFA.

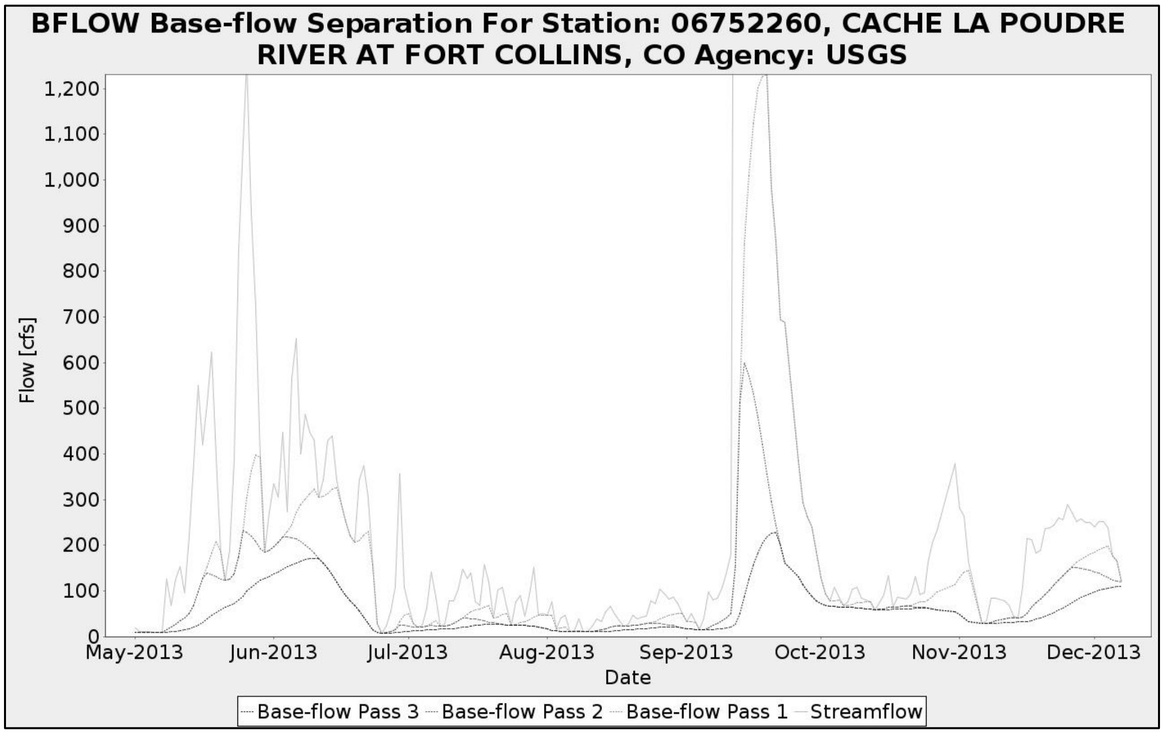


Figure 65: Example CFA's Base-Flow Separation (BFLOW) Tool Result Graph

4.5. DURATION CURVE ANALYSIS

Another approach to stream flow data is the application of duration curves; which statistically rank and graph available flow data. The Flow Duration Curve (FDC) tool in CFA graphs Weibull plotting position ranks of stream flows on a scale of percent exceedence. Graphing flow values in this way allows for a quick visualization of the variability of flow under the different flow regimes and is useful numerically for thresholds such as the flow rate only exceeded 10% of the time in the historical record. The plotting position used in CFA is a tied-rank max. This means for example if there are 3 observations of a flow value of 30cfs that would normally have ranks 13, 14, and 15 the rank of all three observations is re-set to the maximum rank of the ties, in this case rank 15. An example of the output of CFA’s FDC tool is shown below in Figure 66. The black line is the duration curve for the entire period of analysis while there is a light grey line for each annual duration curve in the period of analysis.

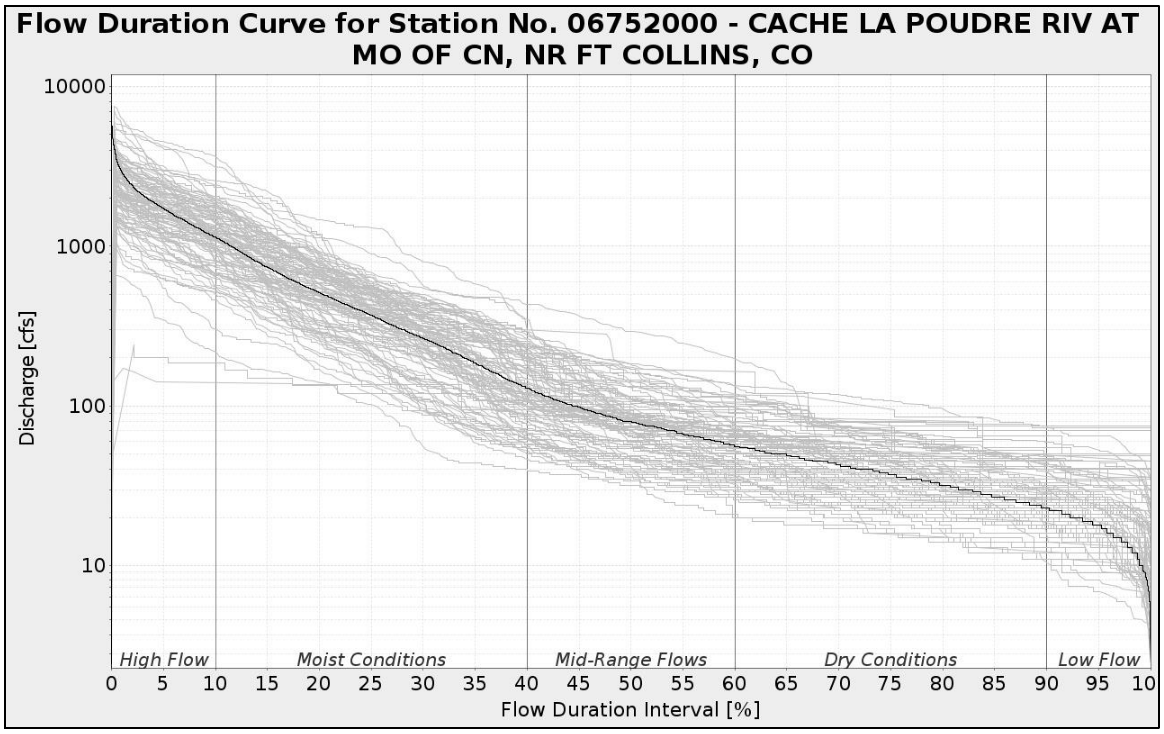


Figure 66: Example CFA's Flow Duration Curve Analysis Tool Result Graph

An extension of the FDC is the Load Duration Curve (LDC) tool in CFA. A LDC is a FDC multiplied by a target water quality concentration level to achieve a load per day value of a particular water quality nutrient. In addition to the LDC itself, observed water quality samples can be graphed as loads (flow * water quality concentration * conversion factors = load). If the observed loads never occur above the LDC line then there is no indication of a water quality problem for that desired target concentration, as shown in Figure 67.

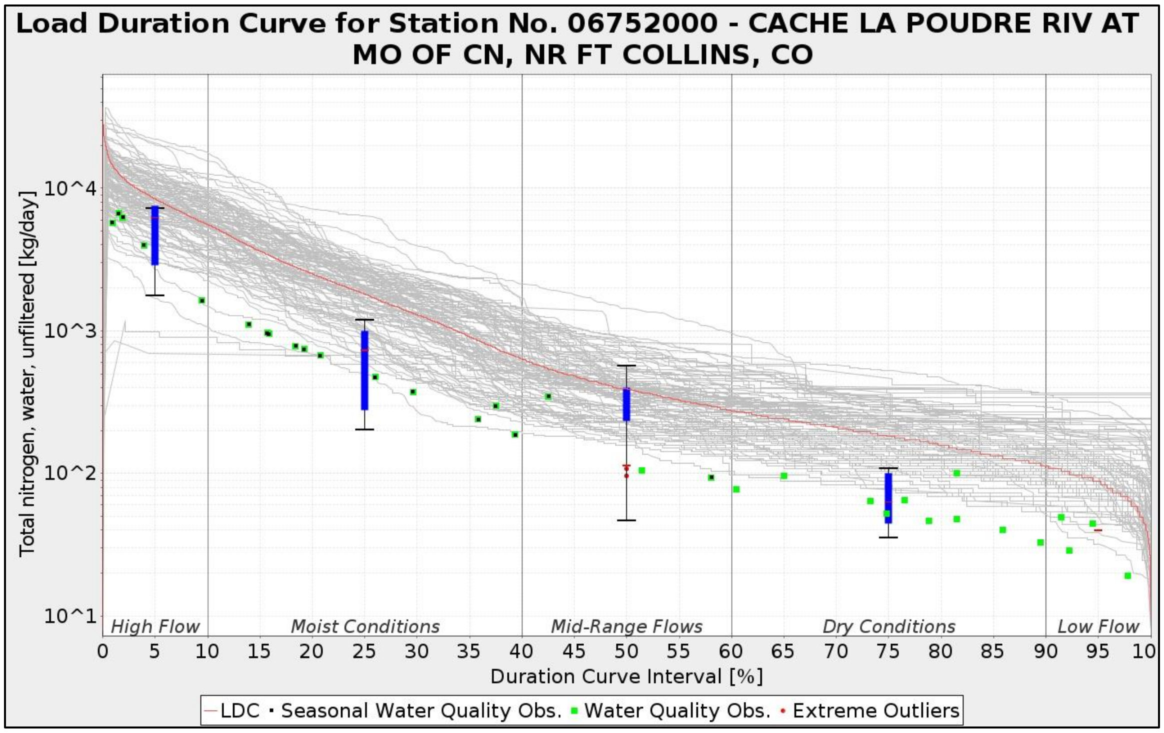


Figure 67: Example CFA's Load Duration Curve Analysis Tool Result Graph 1

If there are observations which exceed the LDC, shown in Figure 68, it can sometimes help determine, based on where the observations exceed the curve, what pollution sources are probable contributors (Cleland 2007, Cleland 2003 and Cleland 2002). Based on the location of these exceedences and the outline provided in (Cleland 2007, Cleland 2003 and Cleland 2002), CFA's LDC dynamically estimates possible nutrient pollution sources based on the location and magnitude of the exceeded values on the graph and reports this back to the user. In addition to the more complex analysis of identifying pollutant sources, LDCs can also be used to identify Total Maximum Daily Loads (TMDLs) for different flow regimes of a river of interest (Cleland 2007). The value of the LDC at a given exceedence is equal to the TMDL for that river, minus a margin of safety, for the specified pollutant and target water quality concentration.

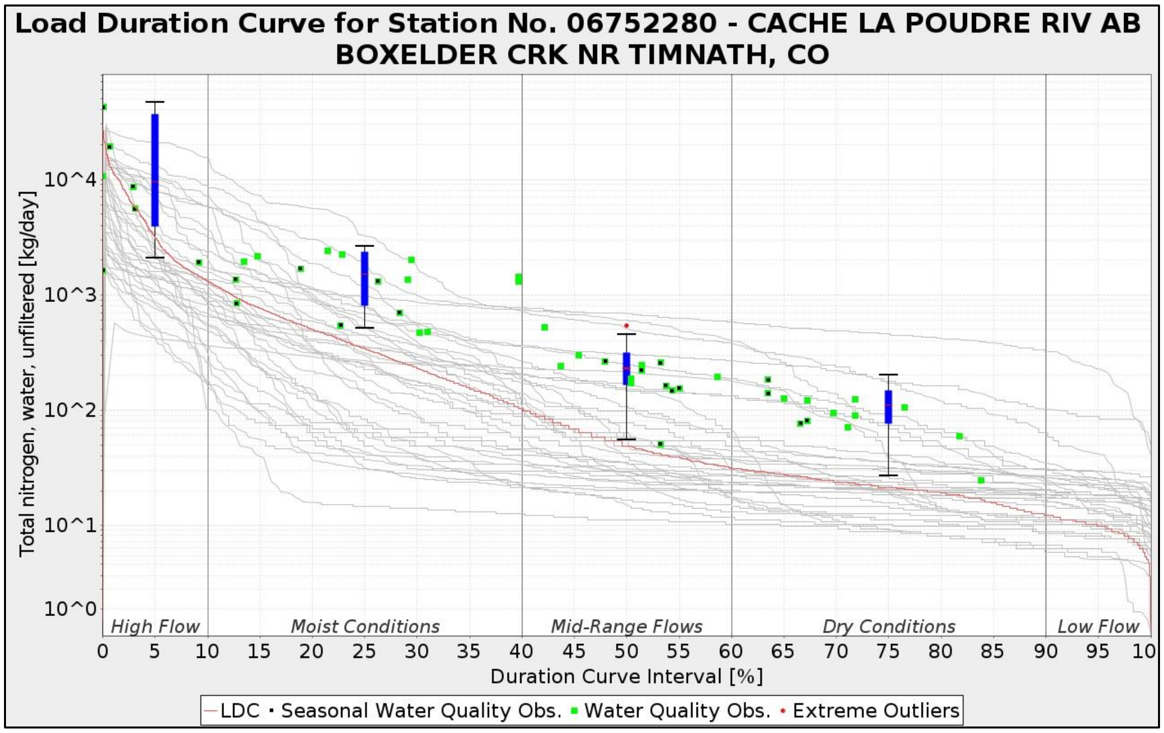


Figure 68: Example CFA's Load Duration Curve Analysis Tool Result Graph 2

4.6. LOAD ESTIMATOR (LOADEST)

Another executable included in CFA is the Load Estimator which is a tool (LOADEST) developed by the U.S. Geological Survey (Runkel et al. 2004). LOADEST is a FORTRAN executable that estimates the amount of constituent loads in streams and rivers given a time series of stream flows and constituent concentrations. Estimation of constituent loads occurs in two steps, the calibration procedure and the estimation procedure, both of which are based on three statistical estimation methods. These methods are Adjusted Maximum Likelihood Estimation (AMLE), Maximum Likelihood Estimation (MLE) and Least Absolute Deviation (LAD). The first two methods are appropriate when the calibration model errors, or residuals, are normally distributed. Of these two, AMLE is best utilized when the calibration data (i.e. stream flow and constituent concentration) are censored. The LAD is an alternative to maximum likelihood estimation when the residuals are not normally distributed.

In the calibration step, known constituent concentrations with corresponding stream flows are used to calibrate LOADEST so that it may be determined which of the preloaded models in LOADEST may best be used for determining the load. Next, in the estimation step, all of the known stream flows are used to estimate loads of constituent for each day. CFA then provides a time series graph, see Figure 69, of the loads estimated by LOADEST. These loads can be determined in either grams, kilograms, pounds or tons. CFA also provides a boxplot and a statistical summary of the estimated loads for the given time period determined by the stream flow data. Finally, if daily stream flow values are available, these daily loads (Figure 69) can be summed in CFA to provide monthly (Figure 70) or even yearly (Figure 70) values of constituent loads in streams and rivers and their corresponding time series and boxplots will be provided.

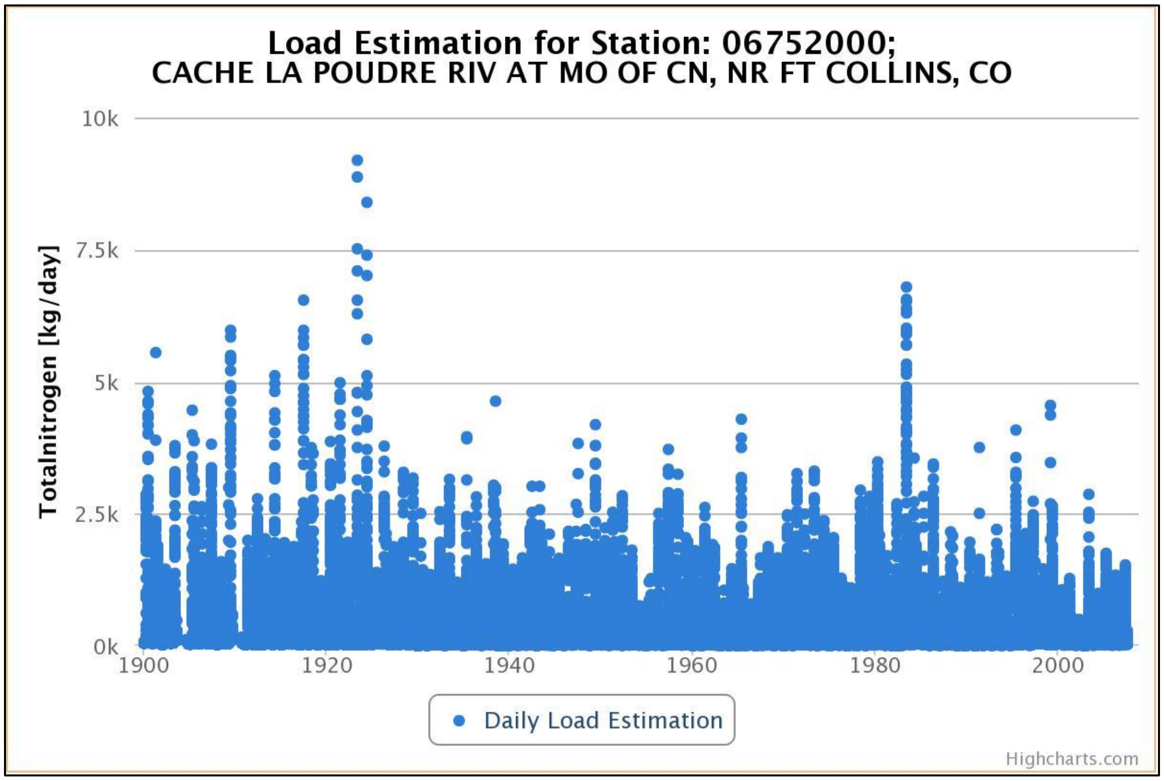


Figure 69: Example CFA's Load Estimator (LOADEST) Analysis Tool Result Daily Graph

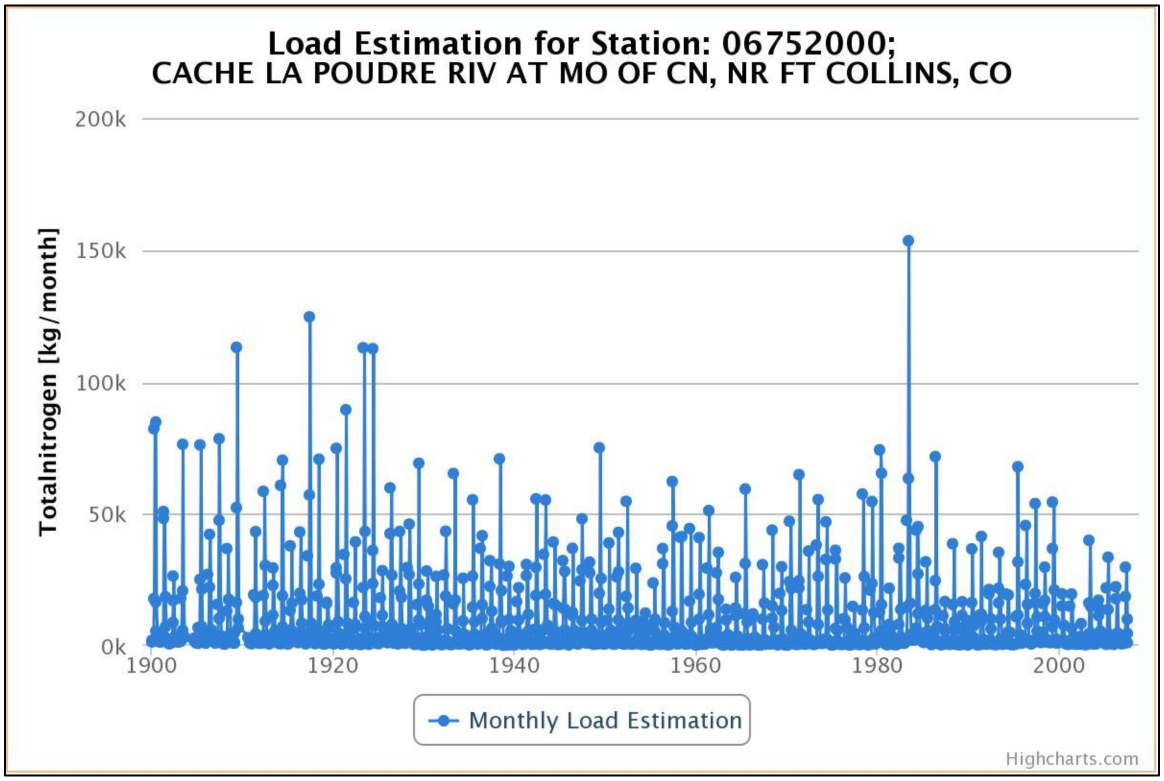


Figure 70: Example CFA's Load Estimator (LOADEST) Analysis Tool Result Monthly Graph

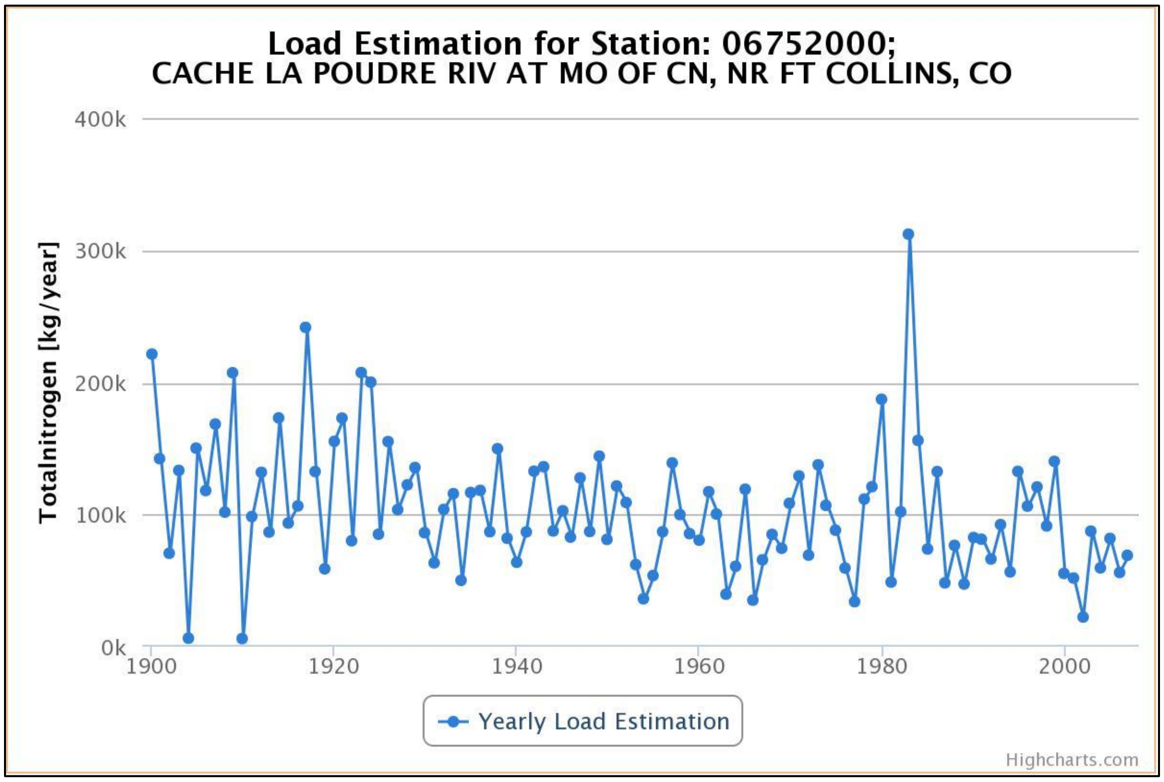


Figure 71: Example CFA's Load Estimator (LOADEST) Analysis Tool Result Annual Graph

5. COMPREHENSIVE FLOW ANALYSIS TECHNICAL MANUAL REFERENCES

Arnold, J. G., P. M. Allen, R. Muttiah, & G. Bernhardt. 1995. "Automated base flow separation and recession analysis techniques." *Ground Water*, 33(6): 1010-1010.

Arnold, J.G. and P.M. Allen. 1999. "Automated methods for estimating baseflow and ground water recharge from streamflow records." *Journal of the American Water Resources Association* 35(2): 411-424.

Cleland, Bruce. August 2002. "TMDL Development from the 'Bottom Up' Part II: Using Duration Curves to Connect the Pieces." *National TMDL Science and Policy 2002.*

Cleland, Bruce. November 2003. "TMDL Development from the 'Bottom Up' - Part III: Duration Curves and Wet-Weather Assessments." *National TMDL Science and Policy*.

Cleland, Bruce. August 2007. "An Approach for Using Load Duration Curves in the Development of TMDLs."*National TMDL Science and Policy*.

David, Olaf, James P. Lyon, Wesley Lloyd, and Kenneth W. Rojas. 2012 "Model Services: REST Specification and Service Signatures."

Interagency Advisory Committee on Water Data (IACWD). 1982. "Guidelines for determining flood flow frequency." *Bulletin No. 17B* (revised and corrected), Hydrology Subcommittee, Washington, D.C.

Lim, Kyong Jae, Bernard A. Engel, Zhenxu Tang, Joongdae Choi, Ki-Sung Kim, Suresh Muthukrishnan, and Dibyajyoti Tripathy. December 2005. "Automated Web GIS Based

Hydrograph Analysis Tool, WHAT." *Journal of the American Water Resources Association*. 1407-1416.

Lloyd, W., O. David, J. Lyon, K.W. Rojas, J.C. Ascough II, T.R. Green, J.R. Carlson. 2012. "The Cloud Services Innovation Platform-Enabling Service-Based Environmental Modelling Using Infrastructure-as-a-Service Cloud Computing." *International Environmental Modelling and Software Society Conference 2012.*

Mishra, Ashok K. and Singh, Vijay P. 2011. "Drought Modeling – A Review." *Journal of Hydrology 403: 157-175.*

Runkel, Robert L., Charles G. Crawford, and Timothy A. Cohn. U.S. Department of the Interior, U.S. Geological Survey. 2004. "Chapter A5: Load Estimator (LOADEST): A FORTRAN Program for Estimating Constituent Loads in Streams and Rivers". *Techniques and Methods Book 4*. Reston, Virginia

Salas, Jose D., Chongjin Fu, Antonino Cancelliere, Dony Dustin, Dennis Bode, Andy Pineda, and Esther Vincent. 2005. "Characterizing the Severity and Risk of Drought in the Poudre River, Colorado." *Journal of Water Resources Planning and Management* 131(5): 383-393.

Salas, Jose D. 1993. "Chapter 19: Analysis and Modeling of Hydrologic Time Series." *The McGraw Hill handbook of hydrology*. D. R. Maidment, ed., McGraw-Hill New York

Sloto, Ronald A., and Michéle Y Crouse. 1996. "HYSEP: A Computer Program For Streamflow Hydrograph Separation and Analysis." U.S. Geological Survey. Lemoyne, Pennsylvania.

Water Resources Council, Hydrology Committee (WRC-HC). 1967. "A Uniform Technique for Determining Flood Flow Frequencies." *Bulletin No. 15*, Washington, D.C.

WineHQ – Run Windows applications on Linux, BSD, Solaris, and Mac OS X, 2012, http://www.winehq.org/

APPENDIX VII: CFA STATE REGIONAL FLOOD SKEWNESS COEFFIECIENT REFERENCES

Table 10: CFA's State Regional Flood Skewness Coefficient References

Alabama:	Olin, D. A. Magnitude and Frequency of Floods in Alabama. Rep. no. 84-4191. U.S. Geological Survey. Web. 5 July 2012. <http://pubs.usgs.gov/wri/1984/4191/report.pdf>.
Alaska:	Curran, Janet H., David F. Meyer, and Gary D. Tasker. Estimating the Magnitude and Frequency of Peak Streamflows for Ungaged Sites on Streams in Alaska and Conterminous Basins in Canada. Rep. no. 03-4188. U.S. Geological Survey. Web. 5 June 2012. <http://pubs.usgs.gov/wri/wri034188/pdf/wri034188_v1.10.pdf>.
Arizona:	Eychaner, James H. ESTIMATION OF MAGNITUDE AND FREQUENCY OF FLOODS IN PIMA COUNTY, ARIZONA, WITH COMPARISONS OF ALTERNATIVE METHODS. Rep. no. 84-4142 U.S. Geological Survey. Web. 5 July 2012. <http://pubs.usgs.gov/wri/1984/4142/report.pdf>.
California:	Parrett, Charles, Andrea Veilleux, and J. R. Stedinger. Regional Skew for California, and Flood Frequency for Selected Sites in the Sacramento-San Joaquin River Basin, Based on Data through Water Year 2006. Rep. no. 2010-5260. U.S. Geological Survey. Web. 9 July 2012. <http://pubs.usgs.gov/sir/2010/5260/pdf/sir20105260.pdf>.
Colorado:	Vaill, J.E., 2000, Analysis of the magnitude and Frequency of Floods in Colorado: U.S. Geological Survey Water Resources Investigations Report 99-4190.
Connecticut:	Ahearn, Elizabeth A. Peak-Flow Frequency Estimates for U.S. Geological Survey Streamflow-Gaging Stations in Connecticut. Rep. no. 03-4196. U.S. Geological Survey. Web. 8 June 2012. <http://pubs.usgs.gov/wri/wri034196/wrir03-4196.pdf>.
Delaware:	Ries III, Kernell G., and Jonathan J.A. Dillow. Magnitude and Frequency of Floods on Nontidal Streams in Delaware. Rep. no. 2006-5146. U.S. Geological Survey. Web. 9 July 2012. <http://pubs.usgs.gov/sir/2006/5146/pdf/sir2006-5146.pdf>.
Florida:	Hammett, K. M., and M. J. DelCharco. Estimating the Magnitude and Frequency of Floods for Streams in West-Central Florida, 2001. Rep. no. 2005-5080. U.S. Geological Survey. Web. 10 July 2012. <https://store.usgs.gov/yimages/PDF/205602.pdf>.
Georgia:	Gotvald, Anthony J., Toby D. Feaster, and J. Curtis Weaver. Magnitude and Frequency of Rural Floods in the Southeastern United States, 2006: Volume 1, Georgia. Rep. no. 2009-5043. U.S. Geological Survey. Web. 10 July 2012. <http://pubs.usgs.gov/sir/2009/5043/pdf/SIR2009_5043_book_508_V2.pdf>.
Hawaii:	Oki, Delwin S., Sarah N. Rosa, and Chui W. Yeung. Flood-Frequency Estimates for Streams on Kauai, Oahu, Molokai, Maui, and Hawaii, State of Hawaii. Rep. no. 2010-5035. U.S. Geological Survey. Web. 10 June 2012. <http://pubs.usgs.gov/sir/2010/5035/sir2010-5035_text.pdf>.

Idaho:	Berenbrock, Charles. Estimating the Magnitude of Peak Flows at Selected Recurrence Intervals for Streams in Idaho. Rep. no. 02-4170. U.S. Geological Survey, 2002. Web. 6 July 2012. <http://id.water.usgs.gov/PDF/wri024170/regression.pdf>.
Illinois	Soong, David T., Audrey L. Ishii, Jennifer B. Sharpe, and Charles F. Avery., 2004, Estimating Flood-Peak Discharge Magnitudes and Frequencies for Rural Streams in Illinois. Rep. no. 2004-5103. U.S. Geological Survey. Web. 10 June 2012. <http://il.water.usgs.gov/pubs/sir2004-5103.pdf>.
Indiana:	Techniques For Estimating Magnitude And Frequency Of Floods On Streams In Indiana. Rep. no. 84-4134. U.S. Geological Survey. Web. 6 July 2012. <http://pubs.usgs.gov/wri/wrir_84-4134/pdf/wrir_84-4134_b.pdf>.
Iowa:	Eash, David A. TECHNIQUES FOR ESTIMATING FLOOD-FREQUENCY DISCHARGES FOR STREAMS IN IOWA. Rep. no. 00-4233. U.S. Geological Survey, 2001. Web. 6 June 2012. <http://ia.water.usgs.gov/pubs/reports/WRIR_00-4233.pdf>.
Kansas:	Rasmussen, Patrick P., and Charles A. Perry. Estimation of Peak Streamflows for Unregulated Rural Streams in Kansas. Rep. no. 00-4079. U.S. Geological Survey, 200. Web. 6 June 2012. <http://ks.water.usgs.gov/pubs/reports/wrir.00-4079.pdf>.
Kentucky:	Hodgkins, Glenn A., and Gary R. Martin. Estimating the Magnitude of Peak Flows for Streams in Kentucky for Selected Recurrence Intervals. Rep. no. 03-4180. U.S. Geological Survey, 2003. Web. 26 June 2012. <http://pubs.usgs.gov/wri/wri034180/pdf/wri034180.pdf>.
Maryland:	Pedersen, Neil J., and Jay G. Sakai. Application of Hydrologic Methods in Maryland. Rep. 3rd ed, 2010. U.S. Geological Survey. Web. 26 June 2012. <http://www.gishydro.umd.edu/HydroPanel/hydrology_panel_report_3rd_edition_final.pdf>.
Massachusetts:	Wandle, Jr., William S. Estimating Peak Discharges of Small, Rural Streams in Massachusetts. Rep. no. 2214. U.S. Geological Survey, n.d. Web. 11 July 2012. <http://pubs.usgs.gov/wsp/2214/report.pdf>.
Michigan:	Croskey, H. M. Estimating Generalized Flood Skew Coefficients for Michigan, 1983. Rep. no. 83-4194. U.S. Geological Survey. Web. 16 June 2012. <http://pubs.usgs.gov/wri/1983/4194/report.pdf>.
Minnesota:	Lorenz, D. L. Generalized Skew Coefficients for Flood-Frequency Analysis in Minnesota. Rep. no. 97-4089. U.S. Geological Survey. Web. 22 June 2012. <http://pubs.usgs.gov/wri/1997/4089/report.pdf>.
Mississippi:	Landers, Mark N., and K. Van Wilson, Jr. FLOOD CHARACTERISTICS OF MISSISSIPPI STREAMS. Rep. no. 91-4037. U.S. Geological Survey. Web. 11 June 2012. <http://pubs.usgs.gov/wri/1991/4037/report.pdf>.
Missouri:	Southard, Rodney E. Estimation of the Magnitude and Frequency of Floods in Urban Basins in Missouri. Rep. no. 2010-5073. U.S. Geological Survey. Web. 25 June 2012. <http://pubs.usgs.gov/sir/2010/5073/pdf/SIR2010-5073.pdf>.

Montana:	Parrett, Charles, and D. R. Johnson. Methods for Estimating Flood Frequency in Montana Based on Data through Water Year 1998. Rep. no. 03-4308. U.S. Geological Survey. Web. 20 June 2012. <http://pubs.usgs.gov/wri/wri03-4308/pdf/wrir03_4308.pdf>.
Nebraska:	Soenksen, Philip J., Lisa D. Miller, Jennifer B. Sharpe, and Jason R. Watton. Peak-Flow Frequency Relations and Evaluation of the Peak-Flow Gaging Network in Nebraska. Rep. no. 99-4032. U.S. Geological Survey. Web. 26 June 2012. <http://content.lib.utah.edu/utils/getfile/collection/wwdl-er/id/273/filename/image>.
New Hampshire:	Olson, Scott A. Estimation of Flood Discharges at Selected Recurrence Intervals for Streams in New Hampshire. Rep. no. 2008-5206. U.S. Geological Survey. Web. 18 June 2012. <http://pubs.usgs.gov/sir/2008/5206/pdf/sir2008-5206.pdf>.
New Jersey:	Watson, Kara M., and Robert D. Schopp. Methodology for Estimation of Flood Magnitude and Frequency for New Jersey Streams. Rep. no. 2009-5167. U.S. Geological Survey. Web. 12 June 2012. <http://pubs.usgs.gov/sir/2009/5167/pdf/sir2009-5167.pdf>.
New York:	Lumia, Richard, and Yvonne H. Baevsky. Development of a Contour Map Showing Generalized Skew Coefficients of Annual Peak Discharges of Rural, Unregulated Streams in New York, Excluding Long Island. Rep. no. 00-4022. U.S. Geological Survey. Web. 11 June 2012. <http://ny.water.usgs.gov/pubs/wri/wri004022/WRIR00-4022.pdf>.
North Carolina:	Pope, Benjamin F., Gary D. Tasker, and Jeane C. Robbins. Estimating the Magnitude and Frequency of Floods in Rural Basins of North Carolina. Rep. no. 01-4207. U.S. Geological Survey. Web. 11 June 2012. <http://nc.water.usgs.gov/reports/wri014207/pdf/report.pdf>.
North Dakota:	Williams-Sether, Tara. Techniques for Estimating Peak-Flow Frequency Relations for North Dakota Streams. Rep. no. 92-4020. U.S. Geological Survey. Web. 22 June 2012. <http://nd.water.usgs.gov/pubs/wri/wri924020/pdf/wri924020.pdf>.
Oklahoma:	Tortorelli, Robert L. Techniques for Estimating Peak-Streamflow Frequency for Unregulated Streams and Streams Regulated by Small Floodwater Retarding Structures in Oklahoma. Rep. no. 97-4202. U.S. Geological Survey. Web. 12 July 2012. <http://pubs.usgs.gov/wri/wri974202/pdf/wri97-4202.pdf>.
Oregon:	Cooper, Richard M. Estimation of Peak Discharges for Rural, Unregulated Streams in Western Oregon. Rep. no. 2005-5116. U.S. Geological Survey. Web. 11 June 2012. <http://pubs.usgs.gov/sir/2005/5116/pdf/sir2005-5116.pdf>.
Rhode Island:	Zarriello, Phillip J., Elizabeth A. Ahearn, and Sarah B. Levin. Magnitude of Flood Flows for Selected Annual Exceedance Probabilities in Rhode Island Through 2010. Rep. no. 2012-5109. U.S. Geological Survey. Web. 10 July 2012. <http://pubs.usgs.gov/sir/2012/5109/pdf/sir2012-5109_report_508.pdf>.

South Carolina:	Feaster, Toby D., and Gary D. Tasker. Techniques for Estimating the Magnitude and Frequency of Floods in Rural Basins of South Carolina, 1999. Rep. no. 02-4140. U.S. Geological Survey. Web. 22 June 2012. <http://pubs.usgs.gov/wri/wri024140/pdf/wrir02-4140.pdf>.
South Dakota:	Sando, Steven K. Techniques for Estimating Peak-Flow Magnitude and Frequency Relations for South Dakota Streams. Rep. no. 98-4055. U.S. Geological Survey. Web. 6 July 2012. <http://pubs.usgs.gov/wri/wri98-4055/pdf/wri98-4055_new.pdf>.
Tennessee:	Wibben, Herman C. Application of the U.S. Geological Survey Rainfall-Runoff Simulation Model to Improve Flood Frequency Estimates on Small Tennessee Streams. Rep. no. 76-120. U.S. Geological Survey. Web. 12 July 2012. <http://pubs.usgs.gov/wri/wri76-120/pdf/wrir_76-120_a.pdf>.
Texas:	Judd, Linda J., William H. Asquith, and Raymond M. Slade, Jr. TECHNIQUES TO ESTIMATE GENERALIZED SKEW COEFFICIENTS OF ANNUAL PEAK STREAMFLOW FOR NATURAL BASINS IN TEXAS. Rep. no. 96-4117. U.S. Geological Survey. Web. 6 June 2012. <http://pubs.usgs.gov/wri/wri964117/pdf/wri96-4117.pdf>.
Utah:	Pomeroy, Christine A., and Aaron J. Timpson. METHODS FOR ESTIMATING MAGNITUDE AND FREQUENCY OF PEAK FLOWS FOR SMALL WATERSHEDS IN UTAH. Rep. no. UT-10.11. U.S. Geological Survey. Web. 26 June 2012. <http://www.udot.utah.gov/main/uconowner.gf?n=15647703282821991>.
Vermont:	Olson, Scott A. Flow-Frequency Characteristics of Vermont Streams. Rep. no. 02-4238. U.S. Geological Survey. Web. 26 June 2012. <http://pubs.usgs.gov/wri/wrir02-4238/wrir02-4238.pdf>.
West Virginia:	Atkins, Jr., John T., Jeffrey B. Wiley, and Katherine S. Paybins. Generalized Skew Coefficients of Annual Peak Flows for Rural, Unregulated Streams in West Virginia. Rep. no. 2008-1304. U.S. Geological Survey. Web. 18 June 2012. <http://pubs.usgs.gov/of/2008/1304/pdf/ofr2008-1304.pdf>.
Wisconsin:	Walker, J. F., and W. R. Krug. Flood-Frequency Characteristics of Wisconsin Streams. Rep. no. 03-4250. U.S. Geological Survey. Web. 19 July 2012. <http://pubs.usgs.gov/wri/wri034250/pdf/wrir-03-4250.pdf>.
Wyoming:	Miller, Kirk A. PEAK-FLOW CHARACTERISTICS OF WYOMING STREAMS. Rep. no. 03-4107. U.S. Geological Survey. Web. 19 June 2012. <http://pubs.usgs.gov/wri/wri034107/pdf/wri034107.pdf>.