12-2015

Evaluation of Conservation Practices through Simulation Modeling and Tool Development

Gurdeep Singh
University of Arkansas, Fayetteville

Evaluation of Conservation Practices through Simulation Modeling and Tool Development

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy in Engineering

By

Gurdeep Singh
Punjab Agricultural University
Bachelor of Technology in Agricultural Engineering, 2010
University of Arkansas
Master of Science in Biological Engineering, 2012

December 2015
University of Arkansas

This dissertation is approved for recommendations to the Graduate Council.

_____	_____
Dr. Dharmendra Saraswat Dissertation Co-director	Dr. Thomas A. Costello Committee Member
_____	_____
Dr. Andrew N. Sharpley Committee Member	Dr. Xuan Shi Committee Member
_____	_____
Dr. Rodney D. Williams Committee Member	Dr. Michele L. Reba Committee Member

Abstract

For the first objective, predicted impacts of selected Mississippi River Basin Initiative (MRBI) conservation practices (CPs) on sediment and nutrient loss were assessed. The study area was L'Anguille River Watershed (LRW), a priority focus watershed of the MRBI program and the simulated CPs were filter strip, critical area planting, grade stabilization structure, irrigation land leveling, irrigation pipeline, irrigation water management, and nutrient management. The Soil and Water Assessment Tool (SWAT) model was calibrated (1998 – 2005) and validated (2006 – 2012) for flow, sediment, total phosphorus (TP), and nitrate-nitrogen (NO3-N) at the Colt site (503 sq. km. drainage area) and for total flow, surface flow, and base flow at the Palestine site (784 sq. km. drainage area). The statistical results for the calibration and validation were found to be satisfactory or better except a few root mean square error – standard deviations ratio (RSR) values for the calibration period. The SWAT model results were predicted from 2013 – 2017 for assessing the performance of CPs. Out of the CPs used in the LRW, critical area planting was the most effective in reducing the predicted nutrient (58% TP and 16% total nitrogen (TN)) and sediment (80%) loads, followed by filter strip, irrigation land leveling, grade stabilization structure, irrigation pipeline, irrigation water management, and nutrient management. Results such as these could help inform watershed planners and policy makers in selecting appropriate CPs that will most effectively bring about desired nutrient and sediment load reductions.

For the second objective, a CP tool was developed with Python programming language for integrating a user-defined target area utilizing either a single or multiple selection criteria with the SWAT model. The tool uses open source packages such as Geospatial Data Abstraction Library (GDAL) and Matplotlib. The tool is standalone and was designed in such a way that it

simulates CPs at the lowest simulation level (hydrological response unit) of the SWAT model by building a new targeting procedure for SWAT applications and decision-making. The tool automates the process for simulating CPs on a target area and analyzing differences between the baseline and CP scenario. The tool was evaluated for the Cache River Watershed (CRW). A target area was selected in the CRW and irrigation land leveling CP was simulated. A 22% decrease in sediment losses, 20% decrease in TP losses, and 12% decrease in TN losses were predicted. The tool provides a quick approach to address the water quality impacts on a specific target area.

For the third objective, the Python-based CP tool developed for objective 2 was further updated to simulate CPs at user-defined locations using an interactive simulation approach. The tool allows the user to select the target area with mouse-clicks in a user-friendly and interactive environment. The LRW located in northeastern Arkansas was used as the test area. A target area was selected interactively in LRW and filter strip and irrigation land leveling CPs were simulated. A 70% decrease in sediment losses, 68% decrease in TP losses, and 47% decrease in TN losses were predicted.

Acknowledgements

I would like to thank Dr. Dharmendra Saraswat for his continuous support during my PhD program. His constructive critical suggestions helped me a lot in improving my research. I would also like to thank my committee members - Drs. Costello, Reba, Sharpley, Shi, and Williams for their feedback and support in successful completion of my dissertation.

I am also thankful to all my friends and family for their love and blessings. Special thanks goes to Adrian, Ben, Eeshan, Faye, Gagan, Jasleen, Mansoor, Pooja, Prathamesh, Priyanka, and Sunny for motivating me whenever needed.

How can I forget Mrs. Linda Pate? Special thanks goes to her for bearing with me for more than five years and listening to my never-ending complaints. Thanks also goes to the Department of Biological and Agricultural Engineering, Graduate School, Arkansas Natural Resources Commission, and Natural Resources Conservation Service for providing financial assistance to pursue my graduate study.

Dedication

To my family and friends

Contents

1. Introduction and Background 1

1.1 Role of Models to Simulate Conservation Practices 1

1.2 Dissertation Problem 2

1.3 Objectives and Hypotheses 6

1.4 Dissertation Organization 7

1.5 References 8

2. Assessment of Predicted Quantitative Impacts of Selected MRBI CPs on Water Quality in L'Anguille River Watershed, Arkansas 11

2.1 Abstract 11

2.2 Introduction 12

2.3 Materials and Methods 14

2.3.1 Study Area 14
2.3.2 SWAT Model Inputs 16
2.3.3 Model Specifics 19
2.3.4 Calibration and Validation 22
2.3.5 CP Scenarios 27

2.4 Results and Discussion 30

2.4.1 Calibration and Validation 30
2.4.2 CP Scenarios 40

2.5 Challenges in Modeling CPs 43

2.5 Summary 46

2.6 Acknowledgements 46

2.7 References 47

3. CP Targeting Tool: A Tool to Aid in CP Plan Development 52

3.1 Abstract 52

3.2 Introduction 52

3.3 Packages for Tool Development 56

3.4 Description of Various Components in the CP Tool 57

3.5 User Involvement and Background Processes for Each Step in the CP Tool. 61

3.6 Evaluation Procedure 66

3.7 Summary 71

3.7 References 75

4. Development of a SWAT Compatible Desktop-Based Interactive Targeting Tool to Simulate CPs on Target Areas 78

4.1 Abstract 78

4.2 Introduction 78

4.3 Tool Components 80

4.4 Evaluation Procedure 81

4.6 Summary 89

4.7 References 90

5. Summary And Recommendations 92

5.1 Synthesis of Chapter Contents 92

5.1.1 Chapter 2 92
5.1.2 Chapter 3 93
5.1.3 Chapter 4 94

5.2 Future Directions 94

Appendix A1. Packages required for installing the CP tool 96

Appendix A2. Python codes for the binary targeted area process in the CP Tool 97

Appendix A3. Python codes for the interactive targeted area process in the CP Tool 118

List of Tables

Table 2.1. Model inputs for the L'Anguille River watershed. 17

Table 2.2. Land use and land cover merged categories for the Center for Advanced Spatial Technologies (CAST) and National Land Cover Datasets (NLCD) layers (Reproduced from Pai et al., 2011). 18

Table 2.3. Wetland related information in CAST land use layers. 20

Table 2.4. Row crops related information in NLCD land use layers. 21

Table 2.5. Measured data sites in L'Anguille River Watershed. 26

Table 2.6. SWAT-Check messages/warnings and potential resolution for the LRW SWAT model. 32

Table 2.7. Top five parameters identified as sensitive for flow, sediment, total P, and nitrate-N. 33

Table 2.8. Parameters tweaked for calibration. 35

Table 2.9. Results for the calibration and validation of the LRW SWAT model. 36

Table 3.1: Land use and land covers in Cache River Watershed. 67

Table 3.2: Area-weighted average sediment, total phosphorus, and total nitrogen losses for the pre and post-CP scenario. 73

Table 4.1: Land Cover in L'Anguille River Watershed (CAST, 2007). 82

Table 4.2: Area-weighted average sediment, total phosphorus, and total nitrogen losses for the pre and post-CP scenario. 88

List of Figures

Figure 2.1. Location of the L'Anguille River Watershed, subwatershed boundaries, and monitored gauges. 15

Figure 2.2. Flowchart depicting the revised calibration process (modified from Engel et al., 2007). 23

Figure 2.3. Graphical comparison of the measured and simulated flow data at the Colt site. 37

Figure 2.4. Graphical comparison of the measured and simulated flow data at the Palestine site. 38

Figure 2.5. Graphical comparison of the measured and simulated water quality data at the Colt site. 39

Figure 2.6. Percentage reductions in pollutant losses due to simulation of CPs. 41

Figure 2.7. Cumulative sediment (top), total P (middle), and total N (bottom) losses by percent of contributing watershed area. 44

Figure 3.1. Layout of the CP tool showing the main menu, the watershed image in the image viewer, and an area for status messages and guidance for the user. 58

Figure 3.2. Flowchart depicting the front-end processes in the CP tool. 59

Figure 3.3. Flowchart depicting the back-end processes in the CP tool. 62

Figure 3.4. Identification of targeted HRUs represented in a matrix form: (i) Target area matrix, (ii) Pre-processed HRU matrix, and (iii) Post-processed HRU matrix. 64

Figure 3.5. The cumulative graph for sediment variation over contributing area before and after simulation of the irrigation land leveling CP in a targeted area of the Cache River watershed. 68

Figure 3.6. The cumulative graph for total phosphorus variation over contributing area before and after simulation of the irrigation land leveling CP in a targeted area of the Cache River watershed. 69

Figure 3.7. The cumulative graph for total nitrogen variation over contributing area before and after simulation of the irrigation land leveling CP in a targeted area of the Cache River watershed. 70

Figure 3.8. Percentage reduction in sediment, TP, and TN losses due to the simulation of the irrigation land leveling CP in the targeted area. 72

Figure 4.1. The cumulative graph for sediment variation over contributing area before and after simulation of CPs in L'Anguille River watershed. 84

Figure 4.2. The cumulative graph for total phosphorus variation over contributing area before and after simulation of CPs in L'Anguille River watershed. 85

Figure 4.3. The cumulative graph for total nitrogen variation over contributing area before and after simulation of CPs in L'Anguille River watershed. 86

Figure 4.4. Percentage reduction in sediment, TP, and TN losses due to the simulation of filter strip and irrigation land leveling. 87

1. INTRODUCTION AND BACKGROUND

1.1 ROLE OF MODELS TO SIMULATE CONSERVATION PRACTICES

Nonpoint source (NPS) pollution is mainly caused by rainfall and can arise from many diffuse sources. Examples of NPS sources are agriculture, forestry, etc. Agricultural sources have also been reported for increasing sediment losses (Costa, 1975) and have caused hypoxia problems in the Gulf of Mexico by contributing more than 70% of delivered nitrogen (N) and phosphorus (P) in the Mississippi River Basin (MRB) (Alexander et al., 2008). To improve water quality, NPS pollution from agricultural sources should be reduced.

Reducing NPS pollutants directly from streams and rivers is expensive and time consuming. As a result, NPS pollutants should be reduced by decreasing the transport of pollutants from their sources into streams and rivers. Generation and delivery of pollutants from agricultural sources can be minimized by implementing CPs (EPA, 2003).

Monitoring CPs requires substantial financial resources ($2.6 billion under 2002 Farm Security and Investment Act) and is time consuming. As a result, impacts of various CPs on the environment are commonly quantified using watershed models (Arabi et al., 2006). Modeling of CPs are increasingly used for assessing the effectiveness of CPs in reducing sediment and nutrient runoff (Santhi et al., 2006). Advances in computer processing power further motivates researchers to use NPS simulation models for assessing the impacts of CPs on water quality. Therefore, engineers, researchers, and other environmental professionals feel the need to review, develop, and improve models for simulating CPs and solve water quality problems (Borah and Bera, 2004; Daniel et al., 2011; Xie et al., 2015).

Models when combined with user-friendly interfaces can help decision makers to make decisions for CPs implementations without getting into the complexity of the models. The model outputs needs to be simplified for addressing problems relating with environment and water resources (Singh and Frevert, 2006). Simple interfaces are required for watershed models to answer CP related questions.

1.2 DISSERTATION PROBLEM

An appropriation of approximately $2.6 billion was made for implementing various CPs to reduce agricultural pollutants under 2002 Farm Security and Investment Act (O'Donnell, 2010). The United States Department of Agriculture (USDA) initiated another program named Conservation Effects Assessment Project (CEAP) in 2003 for quantifying environmental impacts of CPs at national, regional, and watershed scales. The USDA Natural Resources Conservation Service (NRCS) launched another initiative – MRBI in 2009 to control both local problems relating with MRBI water quality and global problems relating with hypoxia in the Gulf of Mexico (Perez and Walker, 2014). The MRBI objective is to assist producers in selected watersheds in the MRB to implement CPs for controling sediment and nutrient runoff (USDA, 2015a).

The MRBI involves 13 states: Arkansas, Kentucky, Illinois, Indiana, Iowa, Louisiana, Minnesota, Mississippi, Missouri, Ohio, South Dakota, Tennessee, and Wisconsin (USDA, 2015a). In Arkansas, the MRBI focus area watersheds (and their 8-digit hydrologic unit codes) for the year 2015 are: Bayou Macon (08050002), Boeuf (08050001), Cache (08020302), Lake Conway-Point Remove (11110203), L'Anguille (08020205), Little River Ditches (08020204), Lower Mississippi-Memphis (08010100), Lower Mississippi-Helena (08020100), Lower Mississippi-Greenville (08030100), Lower St. Francis (08020203), and Strawberry (11010012)

(USDA, 2015b). Long-term impacts of MRBI CPs on pollutant reduction can either be monitored and/or modeled.

Among various watershed models, the SWAT model has been used widely for assessing various environmental and water quality scenarios (https://www.card.iastate.edu/swat_articles/). Apart from being freely available, SWAT has the flexibility to simulate various CPs (Folle et al., 2007). In SWAT, a watershed is divided into subwatersheds, and subwatersheds into hydrological response units (HRUs). The HRUs in SWAT are delineated by defining threshold percentages for land use, soil, and slope (Neitsch et al., 2005). For example, if 5% threshold has been defined for land use, and soybean constitutes less than 5% area in a subwatershed, then soybean HRU gets merged with the nearby dominant HRU. Her et al. (2015) reported that the commonly-used thresholds range from 5% to 15%. Larger thresholds decrease the number of HRUs and decrease computational time. Usage of thresholds results in the loss of spatial information in the model (Gitau, 2003). As a result, smaller thresholds (preferably, 0% for land use, soil, and slope) should be used for water quality assessment and locating CPs as well as assessing their impacts on water quality (Her et al., 2015). Therefore, there is a need to assess water quality impacts of selected MRBI CPs using no thresholds for creating HRUs.

The second problem in this dissertation is related to the tools for simulating CPs accurately on target areas. Attempts have been made in the past for developing tools to assess water quality impacts because of land use changes or CP implementations. However, the tools become ineffective when it comes to spatial targeting components or the ability to simulate CPs. The Automated Geospatial Watershed Assessment (AGWA) tool was developed for integrating landscape information with various process models for assessing watershed impacts. It uses either SWAT or kinematic runoff and erosion (KINEROS) model for conducting targeted land

use change modeling. It provides options to convert a polygon (or a land cover) from one type to the other (Weltz et al., 2011). Kepner et al. (2008) used AGWA with the SWAT modeling option to analyze the impact of land use changes on surface water conditions. The tool allows modification of land cover map and its analysis either by one-at-a-time land use change or through random distribution. The tool does not provide options to change CP parameters in the SWAT model. Further, the tool does not have the capability to overcome non-spatial nature of HRUs in SWAT that prevents site-specific implementation of CPs in the model.

Another tool – Long Term Hydrologic Impact Assessment (LTHIA) – was developed by the Agricultural and Biological Engineering Department at Purdue University to estimate changes in runoff, recharge, and NPS pollution that results from land use changes (LTHIA, 2013). The tool predicts average annual runoff and NPS loads by ingesting land use and soil patterns. However, there is no provision for simulating CPs in this tool.

Houston Engineering, Inc. developed an Ag Best Management Practice (BMP) Assessment and Tracking Tool for providing information on effectiveness of agricultural CPs in the state of Minnesota (ABATT, 2014). The tool includes a comprehensive database of agricultural CPs in Minnesota, a web-based BMP assessment tool, and BMP tracking tool. The tool does report the CP effectiveness values from published literatures; however, it does not simulate CP operations.

Tetra Tech developed a Site Evaluation Tool (SET) for designing sites and evaluating pollutant-loading rates (SET, 2000). The SET is used to check the effectiveness of CPs in meeting the allowable loading rates. The tool uses a CP worksheet to assign structural CPs to the desired drainage area. The tool provides the overall percent reduction in pollutant loads when

CPs are assigned and also compares the annual pollutant loading rates for the site with CPs to the standards. However, SET is an Excel based tool and has no spatial CP component included in it. The CPs cannot be simulated on the intended spatial target area.

The Virginia Tech BMP decision support software helps in treating stormwater runoff with the use of selected CPs (EPA, 2008). There are various CP classes: ponds and basin, infiltration, filtration, wetland, manufactured/proprietary CP, and low impact development; and CP characteristics: contributing drainage area, impervious fraction of CPs contributing drainage area, soils, geologic site constraints, other CP implementation considerations, and performance goals. However, there are no spatial component in the software for modeling CPs.

Singh (2012) developed a geographic information system (GIS) based targeting approach for targeting biofuel crop production on marginal lands in LRW, Arkansas. Because of spatial discontinuity among HRUs, it is likely that HRUs with the same identification number could be located at more than one place in a subwatershed (Gassman et al., 2007; Pai et al., 2012). This limitation comes in the way of targeted placement of CPs while setting up the SWAT model. While the approach developed by Singh (2012) overcame the challenge of spatial discontinuity among HRUs, it could not be implemented in already-developed SWAT model for a particular sub-watershed or a watershed, as the HRUs needed to be delineated following the new approach before model setup.

Building and calibrating a new SWAT model is a rigorous as well as time-consuming process (Hormann et al., 2009). Therefore, use of already-developed models for targeted placement of CPs will enhance the usage of models among the SWAT modeling community. This is not an easy task because it requires not only maintaining spatial continuity among HRUs but also checking placement of CPs on their intended locations. Developing such a capability

outside the SWAT model, in the form of a tool (to be referred subsequently as CP Tool), is expected to encourage watershed planners along with modelers to run various scenario analyses.

The third problem in this dissertation is related to the interactive selection of target area for simulation of CPs. The simplest way of selecting a target area of interest is to define a polygon (with mouse clicks) to mark the project area interactively (Parmenter, 2007). Making the CP tool interactive would be the next step in the advancement of CP targeting tools. Interactivity would allow the user to select the target area interactively and simulate CPs on the selected target area.

1.3 OBJECTIVES AND HYPOTHESES

The following objectives and hypotheses will be tested in this dissertation:

Objective 1: Illustrating the process for the assessment of predicted quantitative impacts of selected CPs on water quality in LRW, Arkansas.

Hypothesis 1: Selected CPs will reduce predicted pollutant losses (sediment, TP, and TN) in LRW, Arkansas.

Objective 2: Development of a SWAT compatible desktop-based CP targeting tool to simulate CPs on target areas.

Hypothesis 2: A SWAT compatible desktop-based CP targeting tool will allow simulation of CPs on a target area using a binary-targeted map and the already-developed SWAT models.

Objective 3: Development of a SWAT compatible desktop-based interactive CP targeting tool to simulate CPs on target areas.

Hypothesis 3: A SWAT compatible desktop-based interactive CP targeting tool will allow simulation of CPs on an interactively-selected target area using the already-developed SWAT models.

1.4 DISSERTATION ORGANIZATION

The dissertation is divided into six chapters. Chapter 1 provides information about the research background, dissertation problem, and objectives and hypotheses. Chapter 2 assesses quantitative impacts of selected CPs on water quality in LRW, Arkansas. The chapter details (i) setting up a SWAT model for LRW followed by its calibration and validation, (ii) incorporating MRBI-recommended CPs in the model, and (iii) simulating selected CPs in the model and analyzing water quality impacts measured through sediment, TP, and TN losses. Chapter 3 describes the development of a SWAT compatible desktop-based CP tool for simulating CPs on target areas. The chapter details tasks relating to (i) developing a custom desktop-based CP targeting tool that allows use of an already-developed SWAT model and permits a target area to be used as an input, (ii) incorporating selected MRBI-recommended CPs along with their characterization options in the tool, (iii) integrating provisions for resolving the non-spatial nature of HRUs by building new targeting procedures for SWAT applications, and (iv) evaluating the tool for CRW. Chapter 4 explains the integration of the interactive component in the CP tool. Chapter 4 details tasks relating to (i) incorporating an interactive targeting component in the CP tool to define a target area for CP simulations, and (ii) evaluating the tool for the LRW. Chapter 5 provides a synthesis of the key results from individual studies and information for future studies. Finally, chapter 6 lists all of the references that have been cited in this dissertation.

1.5 REFERENCES

ABATT. (2014). Ag BMP Assessment and Tracking Tool. Fargo, N.D.: Houston Engineering, Inc. Available at http://agbmp.houstoneng.net/. Accessed 8 December 2015.

Alexander, R. B., Smith, R. A., Schwarz, G. E., Boyer, E. W., Nolan, J. V., & Brakebill, J. W. (2008). Differences in phosphorus and nitrogen delivery to the Gulf of Mexico from the Mississippi River Basin. *Environ. Science & Tech.*, *42*(3), 822-830.

Arabi, M., Govindaraju, R. S., Sophocleous, M., & Koelliker, J. K. (2006). Use of distributed models for watershed management: case studies. *Watershed models. Boca Raton, FL: Taylor and Francis*.

Borah, D. K., & Bera, M. (2004). Watershed-scale hydrologic and nonpoint-source pollution models: Review of applications. *Trans. ASAE*, *47*(3), 789-803.

Costa, J. E. (1975). Effects of agriculture on erosion and sedimentation in the Piedmont Province, Maryland. *Geological Soc. Ame. Bulletin*, *86*(9), 1281-1286.

Daniel, E. B., Camp, J. V., LeBoeuf, E. J., Penrod, J. R., Dobbins, J. P., & Abkowitz, M. D. (2011). Watershed modeling and its applications: A state-of-the-art review. *Open Hydro. J.*, *5*(2), 26-50.

EPA. (2003). National Management Measures to Control Nonpoint Source Pollution from Agriculture. EPA 841-B-03-004.

EPA. (2008). Handbook for developing watershed plans to restore and protect our waters. EPA 841-B-08-002.

Folle, S., Dalzell, B., & Mulla, D. (2007). Evaluation of best management practices (BMPs) in impaired watersheds using the SWAT model. *St. Paul, MN: Minnesota Department of Agriculture*.

Gassman, P. W., Reyes, M. R., Green, C. H., Arnold, & J. G. (2007). The Soil and Water Assessment Tool: Historical development, applications, and future research directions. *Trans. ASABE*, 50(4), 1211-1250.

Gitau, M. 2003. A Quantitative Assessment of BMP Effectiveness for Phosphorus Pollution Control: The Town Brook Watershed, New York. Ph.D. Dissertation. The Pennsylvannia State University, Pennsylvannia, USA, 247 pp.

Her, Y., Frankenberger, J., Chaubey, I., & Srinivasan, R. (2015). Threshold effects in HRU definition of the Soil and Water Assessment Tool. *Trans. ASABE*, 58(2), 367-378.

Hormann, G., Koplin, N., Cai, Q., & Fohrer, N. (2009). Using a simple model as a tool to parameterise the SWAT model of the Xiangxi river in China. *Quaternary International*, 208(1), 116-120.

Kepner, W. G., Semmens, D. J., Hernandez, M., & Goodrich, D. C. (2008). Chapter 15: Evaluating hydrological response to forecasted land-use change. In *North America Land Cover Summit*. Washington, DC: Association of American Geographers.

LTHIA. (2013). Long Term Impact Hydrological Assessment. West Lafayette, In.: Purdue University. Available at https://engineering.purdue.edu/~lthia/. Accessed 8 December 2015.

Neitsch, S., Arnold, J., Kiniry, J., & Williams, J. (2005). Soil and Water Assessment Tool Theoretical Documentation, Version 2005, Blackland Research Center, Grassland, Soil, and Water Research Laboratory, Agricultural Research Service, Temple, TX, 494pp.

O'Donnell, T. K. (2010). *Assessing impacts and targeting of agricultural CPs* (Doctoral dissertation, University of Missouri--Columbia).

Pai, N., Saraswat, D., & Srinivasan, R. (2012). Field_SWAT: a tool for mapping SWAT output to field boundaries. *Computers & Geosciences*, *40*, 175-184.

Parmenter, B. (2007). Creating a smaller data set from a larger dataset – vector data. Tufts university GIS tip sheet. Medford, MA.: Tufts University, Department of Urban and Environmental Policy and Planning.

Perez, M., & Walker, S. (2014). "Improving Water Quality: A Review of the Mississippi River Basin Healthy Watersheds Initiative (MRBI) To Target U.S. Farm Conservation Funds." Working Paper. Washington, DC: World Resources Institute. Available online at: wri.org/ publication/MRBI. Accessed 8 December 2015.

Santhi, C., Srinivasan, R., Arnold, J. G., & Williams, J. R. (2006). A modeling approach to evaluate the impacts of water quality management plans implemented in a watershed in Texas. *Environ. Modelling & Software, 21*(8), 1141-1157.

SET. (2000). Site evaluation tool user guide and documentation for the Lake Maumelle drainage basin, Pulaski County, Arkansas. Pasadena, Ca.: Tetra Tech, Inc. Available at: http://co.pulaski.ar.us/pdf/SETUserGuidanceV14.pdf. Accessed 8 December 2015.

Singh, G. (2012). A watershed scale evaluation of selected second generation biofeedstocks on water quality. M.S. thesis. Fayetteville, Ark.: University of Arkansas, Department of Biological Engineering.

Singh, V. P. & Frevert, D. K. (2006). Watershed models. Boca Raton, Fla.: Taylor and Francis.

USDA Mississippi River Basin Healthy Watersheds Initiative Web site. (2015a). Available at http://www.nrcs.usda.gov/wps/portal/nrcs/detail/ia/programs/landscape/?cid=nrcs142p2_007958. Accessed 8 December 2015..

USDA Mississippi River Basin Healthy Watersheds Initiative Web site. (2015b). Available at http://www.nrcs.usda.gov/wps/portal/nrcs/detailfull/national/home/?cid=stelprdb1048200. Accessed 8 December 2015.

Weltz, M., Jolley, L., Goodrich, D., Boykin, K., Nearing, M., Stone, J., Guertin, P., Hernandez, M., Spaeth, K., Pierson, F., Morris, C., & Kepner, B. (2011). Techniques for assessing the environmental outcomes of CPs applied to rangeland watersheds. *J. Soil and Water Conservation, 66*(5), 154A-162A.

Xie, H., Chen, L., & Shen, Z. (2015). Assessment of Agricultural Best Management Practices Using Models: Current Issues and Future Perspectives. *Water, 7*(3), 1088-1108.

2. Assessment of Predicted Quantitative Impacts of Selected MRBI CPs on Water Quality in L'Anguille River Watershed, Arkansas

2.1 Abstract

Excess nutrients remains a major cause of water body impairment in the United States, including the Gulf of Mexico. In response to the growing hypoxic zone in the Gulf of Mexico and local water quality concerns, the U.S. Department of Agriculture - Natural Resources Conservations Service (USDA-NRCS) launched the MRBI. The initiative provides financial and technical support to farmers to implement CPs. Due to field and watershed monitoring costs, models are commonly used to target appropriate CPs to high-risk areas and to estimate potential water quality improvements with CP adoption. This study evaluates the predicted impacts of several CPs on nutrient and sediment loss at the HRU scale in the LRW, a priority focus watershed of the MRBI program. The SWAT model was set up for the LRW using long-term spatial and temporal datasets (1995 to 2012). Information from multiple land use and land cover (LULC) images was processed using remote sensing methods to incorporate missing land uses. In order to accurately quantify model output and retain all spatial data, no thresholds for land use, soil, or slope were used. The SWAT model was calibrated and validated from 1998 – 2005 and 2006 – 2012, respectively for flow, sediment, TP, and NO3-N. To assess CP performance, sediment, TP, and TN loadings were predicted from 2013 – 2017. Out of the MRBI CPs used in the LRW, critical area planting was predicted to be the most effective in reducing nutrient (58% for TP and 16% for TN) and sediment (80%) loads, followed by filter strip, irrigation land leveling, grade stabilization structure, irrigation pipeline, irrigation water management, and nutrient management. Some of the predicted impacts conflicted with expected CP performance. Results such as these have the potential to help inform watershed planners and policy makers to

select and target appropriate CPs that will most effectively bring about desired nutrient and sediment load reductions. It is critical that CP modeling algorithms be defined carefully so that the predictions reflect impacts accurately.

2.2 INTRODUCTION

Excess nutrients such as N and P continue to impair thousands of waterways leading to surface water impairment in the U.S. (Perez and Walker, 2014). The U.S. Environmental Protection Agency (EPA) in 1996 reported that NPS nutrients were the primary source of concern in 40% of rivers, 50% of lakes, and 60% of estuaries surveyed and listed as impaired (U.S. Environmental Protection Agency, 1996). Approximately 15,000 water bodies are impaired for nutrients, and agricultural sources are listed as one of the top sources for these impairments (Hall, 2012). Agricultural sources in the MRB have been reported to contribute more than 70% of the N and P delivered to the northern Gulf of Mexico and linked with its hypoxic condition (Alexander et al., 2008). Further, reports in the mid-2000's implied that appreciably higher commodity payments ($52.2 billion) than conservation subsidies ($8.5 billion) for farms in the MRB (1995 to 2002), has contributed to greater N and P loads discharging into the Gulf and thereby, an increase in the hypoxic zone (Environmental Working Group, 2006 - http://www.ewg.org/reports/deadzone).

Generation and delivery of agricultural N and P can be minimized by implementation of various CPs (U.S. Department of Agriculture, Natural Resources Conservation Service, 2014; U.S. Environmental Protection Agency, 2003). To support adoption and tracking of these conservation measures, the USDA-NRCS launched the MRBI in 2009, providing $320 million over 5 years (U.S. Department of Agriculture, Natural Resources Conservation Service, 2009, 2014). The MRBI initiatives objective is to help producers in selected MRBI watersheds via

cost-share funding in implementing CPs mainly to control runoff (U.S. Department of Agriculture, Natural Resources Conservation Service, 2015a).

The MRBI concentrated its activities to 41 "focus" watershed in 13 states: Arkansas, Kentucky, Illinois, Indiana, Iowa, Louisiana, Minnesota, Mississippi, Missouri, Ohio, South Dakota, Tennessee, and Wisconsin (U.S. Department of Agriculture, Natural Resources Conservation Service, 2015a). In Arkansas, the focus watersheds (8 digit HUC level) are Bayou Macon (08050002), Boeuf (08050001), Cache (08020302), L'Anguille (08020205), Little River Ditches (08020204), Lower St Francis (08020203), and Point Remove – Lake Conway (11110203) (U.S. Department of Agriculture, Natural Resources Conservation Service, 2015b). The MRBI contains a component to quantify the effectiveness of adopted CPs to reduce nutrient and sediment loadings in each of the focus watersheds, via a three-tiered monitoring strategy (U.S. Department of Agriculture, Natural Resources Conservation Service, 2009). This involves spatially explicit monitoring at edge-of-field, 12, and 8 digit HUC scales. A reliable determination of nutrient and sediment reductions because of land management / use changes at the above-mentioned three scales can be a lengthy process (Meals et al., 2010; Sharpley et al., 2013). Monitoring is costly and labor intensive. NPS models provide a practical alternative to estimate outcomes (Arabi et al., 2006).

Among various watershed models, the SWAT model has seen widespread application in assessing various environmental scenarios (>2200 publications with SWAT; https://www.card.iastate.edu/swat_articles/). SWAT is freely available, has been actively supported and developed, and has the flexibility to simulate various CPs (Folle et al., 2007). The SWAT model divides a watershed into subwatersheds, and subwatersheds into HRUs. The HRUs are delineated by defining threshold percentages for land use, soil, and slope (Neitsch et al.,

2011). Commonly-used thresholds range from 5% to 15% (Her et al., 2015). Thresholds for creating HRUs result in loss of information but may be justified by a reduction in the computational cost (Gitau, 2003). Her et al. (2015) recommended that smaller thresholds (preferably, 0% for land use, soil and slope) should be used for delineating HRUs if the focus of study is (1) water quality assessment or (2) to incorporate location information for CPs.

The overall objective of this study was to assess the predicted quantitative water quality impacts of selected MRBI-recommended CPs for the LRW using SWAT with no thresholds for land use, soil, and slope to create HRUs. Specific objectives were:

1. Parameterizing SWAT for LRW by enhancing the model setup procedure,
2. Calibrating and validating the SWAT model,
3. Incorporating MRBI-recommended CPs in the SWAT model, and
4. Assessing the water quality impacts of selected MRBI CPs on predicted nutrient and sediment loss from LRW.

2.3 Materials and Methods

2.3.1 Study Area

Land use in LRW is predominantly agriculture, with 69.2, 18.9, 5.1, 3.5, and 1.4% in crop, forest, pasture, urban, and water, respectively, and is located in the Mississippi Delta ecoregion of east central Arkansas (Figure 2.1). The LRW drains an area of 2,474 square kilometers in LRW and land covers in LRW are corn, cotton, rice, soybean, and specialty crops (mixed land uses: tomatoes, watermelon, etc.) (CAST, 2007). The Arkansas Natural Resources

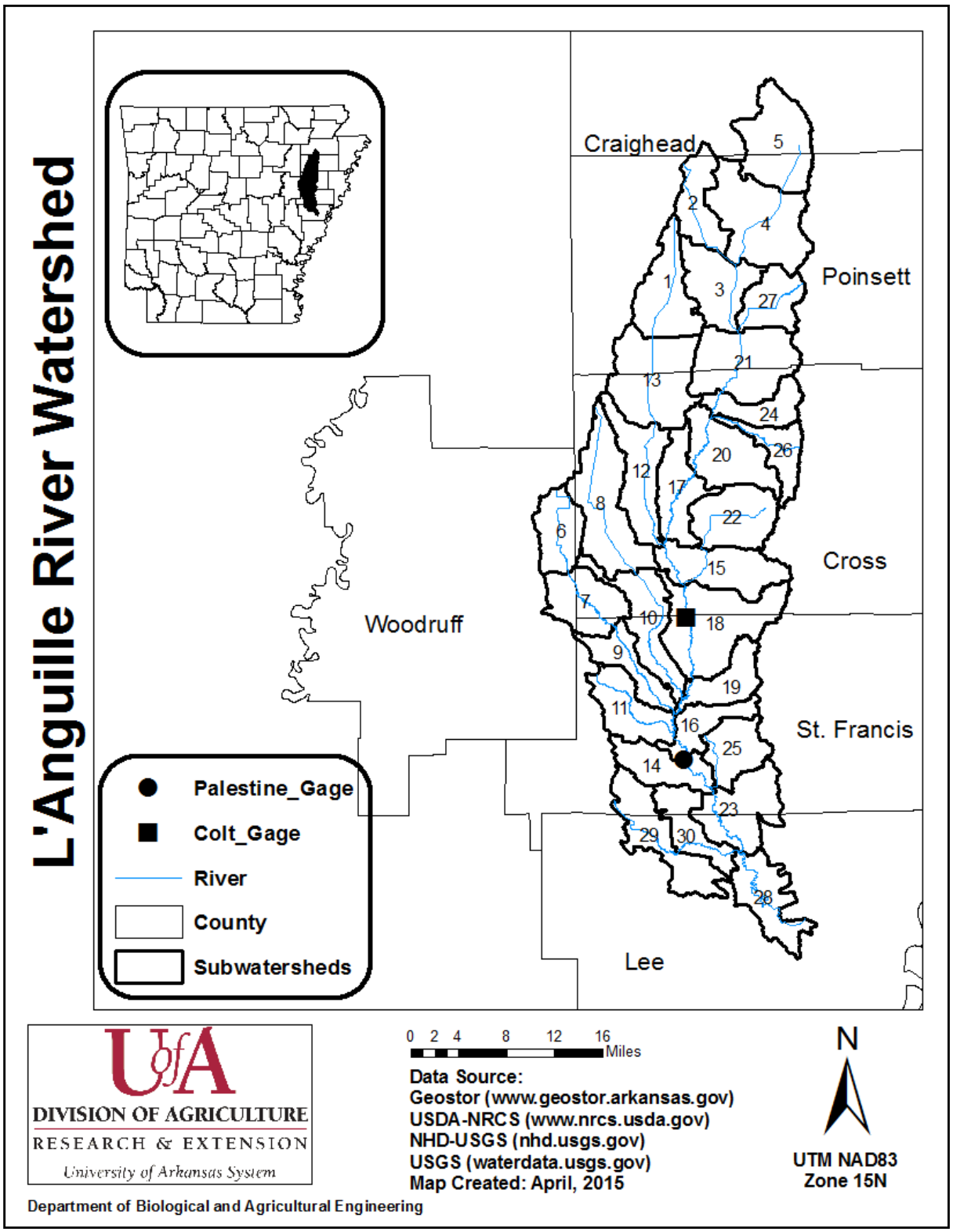

Figure 2.1. Location of the L'Anguille River Watershed, subwatershed boundaries, and monitored gauges.

Commission (ANRC) reported LRW as a priority watershed for nutrients for the 2011-2016 NPS Pollution Management Plan (ANRC, 2012).

2.3.2 SWAT Model Inputs

Input datasets were downloaded in the UTM-Zone 15N (North American Datum 1983 (NAD83) Universal Transverse Mercator Zone 15N) projection system. Various SWAT inputs along with their data sources and resolutions are shown in Table (2.1). The SWAT model (SWAT2012: Revision 627) for the LRW was developed with ArcSWAT 2012.10_1.15 (July 2014) interface that uses ArcGIS 10.1 version. The SSURGO database was used for obtaining soil information, as it is the most comprehensive soil database available for Arkansas (Pai et al., 2011). Studies have indicated better hydrological responses by using NEXRAD precipitation data in SWAT (Kalin and Hantush, 2006; Moon et al., 2004). As the NEXRAD-SWAT tool (for processing NEXRAD datasets) was not compatible with ArcGIS 10.1, the NEXRAD dataset was processed using ArcGIS 9.3.

Multiple temporal land uses have indicated improved spatial and temporal hydrological responses from the model (Pai and Saraswat, 2011). Moreover, Chiang et al. (2010) reported that the land use changes could mask water quality improvements resulting from implementation of CPs in a watershed. Thus, land use and land cover categories included in data from the Center for Advanced Spatial Technologies (CAST) and National Land Cover Dataset (NLCD) (Gorham and Tullis, 2007; Homer et al., 2004) were merged following an approach detailed by Pai et al. (2011) to obtain a common land use classification for all the LULC data layers (Table 2.2).

The land-use layers in NLCD have wetland related information, as opposed to CAST land-use layers. It is known that wetlands control and alter the hydrologic cycle of a watershed

Table 2.1. Model inputs for the L'Anguille River watershed.

Input Data	Description
Elevation	10-m resolution (geostor.arkansas.gov; 2006)
Slope	Classes (0-3%, 3-8%, 8-12%, and >12%) (geostor.arkansas.gov; 2006)
Watershed Boundary	1:24000 (datagateway.nrcs.usda.gov; 2014)
Stream Network	1:24000 (nhd.usgs.gov; 2014)
Soil	SSURGO (datagateway.nrcs.usda.gov; 2005)
Land Use	28.5m (www.cast.uark.edu; 1999, 2004, 2006) 30m (www.mlrc.gov; 1995, 2001)
Meteorological	Rain gage (gis.ncdc.noaa.gov; 1995-2012) and NEXRAD (noaa.gov; Apr1996-2012)
Point Source	Municipal and industrial (http://www.adeq.state.ar.us/; 1992-2012)
Management Practices	Row crops, pasture, forestry, and urban practices (http://www.uaex.edu/farm-ranch/crops-commercial-horticulture/verification.aspx; 1995-2012)

Table 2.2. Land use and land cover merged categories for the Center for Advanced Spatial Technologies (CAST) and National Land Cover Datasets (NLCD) layers (Reproduced from Pai et al., 2011).

Agency (Year)	Categories	Category Name	Merged Name
CAST (1999, 2004, 2006)	11, 14	Intensity 1 and Urban (other)	Urban low intensity
	12, 13	Intensity 2 and Intensity 3	Urban high intensity
	100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128	Various types of trees (oak, pine, etc.)	Forest
	209, 210	Warm season and cool season grasses	Pasture
NLCD (1992, 2001)	21, 22, 85	Low/High residential or recreational	Urban low intensity
	23, 24	Commercial, industrial, transportation	Urban high intensity
	41, 42, 43	Deciduous, evergreen, mixed	Forest

(Martinez-Martinez et al., 2014), store floodwater, and temporarily retains or filters suspended solids and nutrients (Kannan et al., 2014). Because of the large area of wetlands (about 10% of LRW area) and their importance in nutrient flows in the LRW, it was decided to simulate wetland processes in SWAT. For this purpose, wetland information in CAST data layers was incorporated from NLCD data through image processing techniques. Wetlands classified in 1992 NLCD layer (9% of the watershed area) and 2001 NLCD layer (10% of the watershed area) occupied more or less the same spatial area. Wetland information in the 1999 CAST layer was incorporated from the 1992 NLCD layer, and in the 2004 & 2006 CAST layers was incorporated from the 2001 NLCD layer. The percentage of watershed area occupied by various land uses in CAST layers before and after the wetland incorporation is shown in Table (2.3). Wetlands were simulated using the SWAT default parameters for land use category "WETL".

NLCD 1992 and 2001 layers do not have specific row crop information (such as corn, soybean, etc.); rather it has a general category called "row crops". As a result, LULC data was further enhanced through incorporation of specific row crops information in NLCD layers from cropland data layers (CDL). Specifically, NLCD 2001 layer was split into specific crops with the CDL 2001 layer (Table 2.4). As there are no CDL layers available for 1992, row crop category in the NLCD 1992 layer was not split.

2.3.3 MODEL SPECIFICS

A user-defined approach was used in this study to generate subwatershed boundaries that match the 12-digit HUC boundaries as defined by the USGS. There are no available guidelines specifying what thresholds for land use, soil, and slope, should be selected for water quality assessment studies. However, as per the recommendation of Her et al. (2015), the model was

Table 2.3. Wetland related information in CAST land use layers.

Land Use Layers	Land Use Types	Before Wetland Incorporation (% of watershed area)	After Wetland Incorporation (% of watershed area)	Difference (% of watershed area)
1999 CAST	Barren	50	49	1
	Forest	18	11	7
	Wetland	0	9	-9
	Soybean	26	27	1
2004 CAST	Barren	62	62	1
	Forest	21	12	9
	Wetland	0	10	-10
2006 CAST	Forest	21	12	9
	Wetland	0	10	-10
	Soybean	44	43	1

Table 2.4. Row crops related information in NLCD land use layers.

2001 NLCD Land Use Types	Before Row Crops Splitting (% of watershed area)	After Row Crops Splitting (% of watershed area)	Difference (% of watershed area)
Urban	5	6	-1
Barren	0	2	-2
Forest	7	8	-1
Agricultural	73	11	62
Soybean	0	27	-27
Rice	0	25	-25
Cotton	0	6	-6

setup with 0% thresholds for land use, soil, and slope in order to not lose any spatial information in the watershed and to accurately estimate water quality impacts of implemented CPs. Use of no thresholds resulted in 10,561 HRUs that were simulated using the supercomputer facility at the University of Arkansas and a 64-bit processor with 16 GB Random Access Memory (RAM) memory. The text input and output files of SWAT model were transferred to the supercomputing facility. The Secure Shell Client (SSH) was used to connect the supercomputing nodes to the desktop computer. SWAT codes were converted into linux compatible form. The SWAT model was run with the help of a Portable Batch System (PBS) file. Calibration at the supercomputing facility reduced the SWAT run time to one-third of the time required on the desktop computer.

The SWAT model was setup from 1995 to 2012 with 3 years as a warm-up period. The calibration period was from 1998 to 2005 and the validation period from 2006 to 2012. The model was calibrated and validated for flow, sediment, TP, and NO3-N. A flowchart for the calibration methodology developed by Santhi et al. (2001) and revised by Engel et al. (2007) was further revised in this study (Figure 2.2).

2.3.4 CALIBRATION AND VALIDATION

The soil P routines in SWAT model simulate three inorganic (active, solution, and stable), and two organic (fresh and humic) pools (Neitsch et al., 2011). As soil P data (site-specific) for watersheds is not available, fixed amounts of soil P pools for each HRU are initialized in SWAT. Though inclusion of a warm-up period before model calibration tends to stabilize nutrient pools, selection of initial values for various nutrient pools has not been discussed much in the literature. Vadas and White (2010) provided an approach to initialize

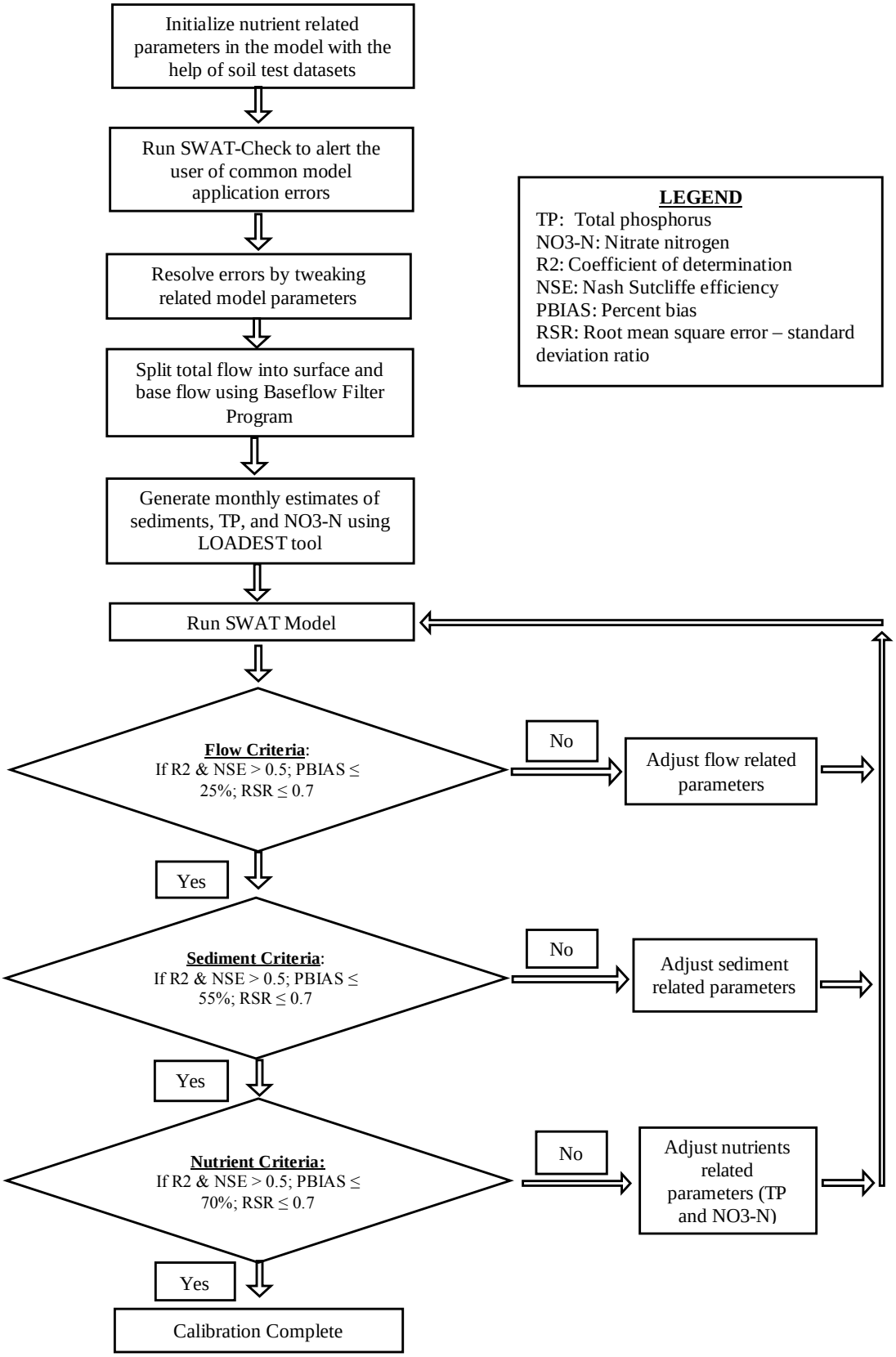

Figure 2.2. Flowchart depicting the revised calibration process (modified from Engel et al., 2007).

solution P and P sorption coefficient (PSC) parameters in SWAT with existing soil chemical data for resulting in a more accurate initialization of the nutrient dynamics for watersheds.

A long-term soil test dataset was acquired for three counties (Craighead, Cross, and Lee) within the LRW from the University of Arkansas Soil Testing and Research Laboratory. The soil dataset consisted of sample measurements for the following parameters: pH, P, potassium (K), calcium (Ca), magnesium (Mg), sodium (Na), sulfur (S), iron (Fe), manganese (Mn), copper (Cu), zinc (Zn), NO3-N, organic matter (OM), and electric conductivity (EC) (Espinoza et al., 2006). As the exact location of samples were not available, the data was aggregated on a county basis and it was assumed to be representative of all of the soils in subwatersheds in that county.

As Mehlich-3 soil test method was used in the soil test P values (Espinoza et al., 2007; Mehlich, 1984), Vadas and White (2010) reported that the solution P (Sol_P) pool of SWAT could be initialized as 50% of Mehlich-3 extractable soil P concentrations. Using this approach, final Sol_P values for the Craighead (subwatershed 5), Cross (subwatersheds 7, 8, 12, 15, 17, 20, 22, 24, and 26), and Lee (subwatersheds 28, 29, and 30) counties were determined to be 37.4, 22.2, and 34.2 mg/kg, respectively.

Another parameter – P sorption coefficient (PSC) which is a watershed-scale parameter was estimated based on equations presented in Sharpley et al. (1984). These equations provided PSC values for calcareous, highly weathered, and slightly weathered soils. In LRW, three major soil group present were Henry, Calloway, and Zachary. The suborder of Henry, Calloway, and Zachary were Aqualfs, Udalfs, and Aqualfs, respectively. All these suborders fall within the Alfisols order that is highly weathered. For highly weathered soils, the PSC was estimated using the following equation:

$$PSC = -0.053 \times \ln(\%clay) + 0.001 \times (Solution\ P, mg\ kg^{-1}) - 0.029 \times (\%Organic\ C) + 0.42 \quad \text{(1)}$$

The %Clay and %Organic C in the top layer was averaged for the top three soils groups obtained from the SSURGO datasets. The Sol_P value was obtained by averaging the values obtained for Craighead, Cross, and Lee Counties, individually. Based on these values, the estimated PSC value came out to be 0.29 and was used in the SWAT model for initializing P pools.

A standalone tool, SWAT-Check, was used to read selected SWAT outputs and to alert the user of common model application errors (White et al., 2014). SWAT-Check was used before and during the calibration process to resolve potential warnings in the model such as with the hydrological and N cycle processes. Latin Hypercube sampling and the one factor-at-a-time methods were used to conduct sensitivity analysis. The SWAT model was calibrated and validated at a monthly time-step for Colt and Palestine USGS sites. Manual calibration was used to adjust parameters to better represent measured data. The objective functions used to optimize flow, sediment, TP, and NO3-N were coefficient of determination (R^2), Nash-Sutcliffe efficiency (NSE), percent bias (PBIAS), and root mean square error-standard deviations ratio (RSR). The established model performance criteria proposed by Santhi et al. (2001) for R^2 and Moriasi et al. (2007) for NSE, PBIAS, and RSR were used to evaluate the model.

The measured datasets for flow, sediment, TP, and NO3-N were obtained from the USGS website (waterdata.usgs.gov/nwis). Information about the drainage area and data duration at Colt and Palestine are reported in Table (2.5). Flow data was split into surface and base flow using the base flow filter program. Load Estimator (LOADEST) tool was used to obtain continuous monthly values for sediment, TP, and NO3-N at the Colt site. Adjusted maximum likelihood

Table 2.5. Measured data sites in L'Anguille River Watershed.

Gage No.	Location	Drainage Area (sq. km)	Agency Data Providing	Flow Data Duration	Water Quality Data Duration
07047942	Colt	503	USGS (http://www.usgs.gov/)	1995 – 2012	1995 – 2012
07047950	Palestine	784	USGS (http://www.usgs.gov/)	1998 – 2012	1998 – 2012

estimation (AMLE) was used in LOADEST as it assumes that the samples are normally distributed with a constant variance, and generates a nearly unbiased estimates of instantaneous load even for the censored data (data censoring occurs when there is one or more observations with constituent concentrations less than the laboratory detection limit) (Runkel et al., 2004). Several assumptions were made during the LOADEST run. PRTOPT (estimated values print option) was set to 1 so that the individual load estimates were written to a separate output file. SEOPT (standard error option) was set to 3 indicating that the exact standard errors were computed for the AMLE load estimates. LDOPT (load option) was set to 2 to get the load calculations for each month in the estimation file. MODNO (model number) was set to 0 so that the best regression model could be automatically selected based on Akaike Information criteria. UCFLAG (concentration unit flag) was set to 1 since the input concentrations in the calibration file were in mg/l. ULFLAG (load unit flag) was set to 1 to request the output in kg/day. The LOADEST tool was not used for obtaining pollutant estimates at a monthly time step at Palestine due to an insufficient number of water quality samples (only seven) available from October 1998 to December 2012.

2.3.5 CP SCENARIOS

Based on the ability of the SWAT model to simulate CPs, seven MRBI-recommended CPs were simulated in the SWAT model developed for the LRW: filter strip, critical area planting, grade stabilization structure, irrigation land leveling, irrigation pipeline, irrigation water management, and nutrient management. Selected CPs were simulated by changing relevant parameters in SWAT to represent specific processes relating to the CPs in the model to assess sediment, TP, and TN impacts.

A filter strip represents a strip of permanent vegetation that is established at the edge or around the perimeter of a field (NRCS CP 393; http://www.nrcs.usda.gov/Internet/FSE_DOCUMENTS/stelprdb1241319.pdf). This practice applies to cropland and grazing lands. Lacking specific information from the LRW, filter strips were simulated in the SWAT model using the default parameters for the in-built filter strip module in ArcSWAT (i.e. FILTER_RATIO = 40, FILTER_CON = 0.5, and FILTER_CH = 0). Use of the filter strip module results in the creation of operation files in the SWAT model.

Critical area planting represents establishing permanent vegetation on sites that have high erosion rates or have physical, chemical or biological conditions preventing establishment of vegetation with normal practices (NRCS CP 342; http://www.nrcs.usda.gov/Internet/FSE_DOCUMENTS/stelprdb1241316.pdf). Critical area planting was simulated by replacing barren lands (identified from the land use and land cover layer) with pasture areas in good condition (Santhi et al., 2006).

A grade stabilization structure represents a structure controlling the grade and head cutting in channels (NRCS CP 410; http://www.nrcs.usda.gov/Internet/FSE_DOCUMENTS/stelprdb1263175.pdf). Grade stabilization structures were simulated by defining slope steepness of channels [(CH_S(2)] parameter in SWAT as 20% of the actual slope (Santhi et al., 2006).

Irrigation land leveling represents reshaping irrigated land surfaces to planned grades (NRCS CP 464; http://www.nrcs.usda.gov/Internet/FSE_DOCUMENTS/stelprdb1263175.pdf). This practice was applied to all lands where irrigation has been applied. Irrigation land leveling was simulated by reducing the HRU slope (reducing HRU_SLP parameter by 10%) and slope length (reducing SLSUBBSN parameter by one-tenth of the default value) (Kannan et al., 2011).

An irrigation pipeline represents a pipeline installed to convey water for storage or application, as part of an irrigation water system (NRCS CP 430; http://www.nrcs.usda.gov/Internet/FSE_DOCUMENTS/stelprdb1046882.pdf). This practice was also applied to all lands where irrigation has been applied. Irrigation pipeline was simulated by increasing the conveyance efficiency of HRU from 0 to 5 (IRR_EFM in the management operation table) (Kannan et al., 2011).

Irrigation water management represents controlling the rate, amount and timing of irrigation water in an efficient manner (NRCS CP 449; http://www.nrcs.usda.gov/Internet/FSE_DOCUMENTS/stelprdb1263179.pdf). Similar to irrigation land leveling and irrigation pipeline, this practice was also applied to all lands where irrigation has been applied. Irrigation water management was simulated by adjusting the water volume required for irrigation with respect to the seasonal total rainfall received (planting to harvest date) (Kannan et al., 2011). The water application rate was adjusted based on the quantity and the timing of rainfall using the method described by Kannan et al. (2011).

Nutrient management represents managing the amount (rate), source, placement (method of application), and timing of plant nutrients and soil amendments (NRCS CP 590; http://www.nrcs.usda.gov/Internet/FSE_DOCUMENTS/stelprdb1046896.pdf). This practice was applied to all lands where plant nutrients and soil amendments were applied. Nutrient management was simulated by simply reducing fertilizer inputs by 50% of the pre-CP amount. It was felt that this would reflect a more restrictive application of N and P in the watershed.

Simulations were run for five years (2013 to 2017) past the model calibration and validation period. The statistical weather calculator available in SWAT was used for preparing

the projected weather data files. The formula used for calculating pollutant reduction due to CP implementation is shown in equation (2).

$$\boldsymbol{Pollutant\ reduction}\ (\%) = \left(\frac{\boldsymbol{pollutant\ load\ from\ baseline\ scenario - pollutant\ load\ from\ CP\ scenario}}{\boldsymbol{pollutant\ load\ from\ baseline\ scenario}}\right) * \mathbf{100} \quad \ldots\ldots (2)$$

2.4 Results and Discussion

2.4.1 Calibration and Validation

To evaluate the initial model performance, an overall annual water balance was calculated at the watershed scale using the basin level output (output.std). The water balance was calculated with the help of equation (3).

$$\mathbf{Precipitation} = \mathbf{Surface\ runoff} + \mathbf{Groundwater} + \mathbf{Evapotranspiration}\ (\mathbf{ET}) + \mathbf{Deep\ Aquifer\ Recharge} + \mathbf{Water\ consumed\ by\ plants} \quad \ldots\ldots (3)$$

To evaluate water balance, both sides of the above equation were expected to be similar over a long simulation period. This was verified using simulations from 1998 to 2005 (calibration time period) at an annual scale. The sum of surface runoff (342.72 mm), groundwater (165 mm), evapotranspiration (507.5 mm), aquifer recharge (121.71 mm), water in soil and consumed by plants (4.06 mm) accounted for ~ 100 % of the precipitation (1145 mm); hence, the water balance of the watershed was considered satisfactory. The sum of simulated groundwater (165 mm) and lateral (9.88 mm) flow contributed 37% of the total water yield (463 mm) which is similar to the base flow estimated at the USGS gage at Colt (38%). Precipitation in

the watershed ranged from 1105 mm to 1275 mm per year against an average annual precipitation of 1141 mm.

SWAT-Check warnings as well as their potential resolutions are reported in Table (2.6). The warnings were related to the hydrology, N cycle, plant growth, and point sources sections of the SWAT-Check tool. Once warning issues were resolved, the sensitivity analysis was conducted to identify parameters that can be used for model calibration. The five most sensitive parameters for flow, sediment, TP, and NO3-N are listed in Table (2.7). Some parameters (not identified as sensitive parameters) were selected to make a better fit for the measured and simulated data (Santhi et al., 2001).

CN2 affects the overland flow and was ranked as the most sensitive parameter for flow. ESCO was another sensitive parameter for flow which was mainly because LRW receives higher solar radiation. Maringanti (2008) also identified CN2 and ESCO as the sensitive parameters for flow. USLE_P was the most sensitive parameter for sediment which indicated that changes in land use practice factor would affect sediment losses. USLE_C was another sensitive parameter affecting sediment indicating that a change in crop management factors would affect sediment losses. The SPCON, SPEXP, and CH_N2 parameters which impact the channel processes also affected sediment losses indicating that sediment losses in LRW were affected by both overland and channel processes.

PHOSKD was the most sensitive parameter affecting TP. Similar to CN2, PHOSKD was also related to the overland process. As a result, it was observed that the overland processes mostly impacted flow, sediment, and TP in LRW. RCHRG_DP was the most sensitive parameter for NO3-N which was expected as the movement of NO3-N is mainly an underground process.

Table 2.6. SWAT-Check messages/warnings and potential resolution for the LRW SWAT model.

Warnings	Resolution
Section 1: Hydrology	
Water yield may be excessive	This warning appears mostly when the evapotranspiration simulated in the uncalibrated model is low. Evapotranspiration related parameters such as ESCO should be modified during calibration to resolve this warning.
Section 3: Nitrogen Cycle	
Denitrification is zero, consider decreasing SDNCO: (Denitrification threshold water content)	This warning appears when the parameters dictating the start of denitrification (SDNCO and CDN) are zero (default value for these parameters are zero in the model). As a result, these two denitrification parameters should be adjusted during calibration to initiate the process of denitrification in the model.
Section 5: Plant Growth	
Unusually low phosphorus stress	The models nutrient balance was functioning properly and as P was added during the crop management practices, this warning was dismissed (Mike White, agricultural engineer – USDA, personal communication, May 2014).
Section 9: Point Sources	
Inlet/point sources contribute flow but not phosphorus	This warning appears when no P loading is reported in the point source data received from the relevant agency (in this case ADEQ and EPA).
Inlet/point sources contribute flow but not nitrogen	This warning appears when no N loading is reported in the point source data received from the relevant agency (in this case ADEQ and EPA).

Table 2.7. Top five parameters identified as sensitive for flow, sediment, total P, and nitrate-N.

Variable	1	2	3	4	5
Flow	CN2	ESCO	SOL_AWC	GWQMN	ALPHA_BF
Sediment	USLE_P	SPCON	USLE_C	SPEXP	CH_N2
Total P	PHOSKD	PPERCO	SURLAG	SOL_Z	SHALLST_N
Nitrate-N	RCHRG_DP	NPERCO	CH_K2	ALPHA_BF	SOL_NO3

Parameters tweaked for calibration and validation are shown in Table (2.8). The parameters were tweaked in such a way that the parameter value lies within the recommended range for the suggested parameter. The statistical calibration and validation results at Colt and Palestine sites are shown in Table (2.9) and the temporal results at both the sites are shown in Figures (2.3-2.5). The R^2 and NSE ranged from [0.53 to 0.70] and [0.50 to 0.66], respectively, at Colt; and [0.54 to 0.58] and [0.51 to 0.54], respectively, at Palestine for the calibration period. The R^2 and NSE ranged from [0.69 to 0.84] and [0.64 to 0.72], respectively, at Colt; and, [0.77 to 0.80] and [0.61 to 0.72], respectively, at Palestine for the validation period. The R^2, NSE, and PBIAS were satisfactory or better indicating satisfactory goodness of fit; however, RSR was unsatisfactory during the calibration period for sediment, TP and NO3-N. Since both PBIAS and RSR are error indexes, and PBIAS was satisfactory or better for sediment, TP, and NO3-N, calibration results were assumed as satisfactory.

The percent bias (PBIAS) statistics indicated some overprediction for total flow during the calibration period (negative biases) and underprediction during the validation period (positive biases) at Colt site, which could also be seen in Figure (2.3). According to Figure (2.3), there was high underprediction for total flow during March 2002, and high overprediction during October 1998 and 1999, June 2000, and May 2004. Total, surface, and base flow was overpredicted during the calibration and validation period (negative biases) at Palestine Site. SWAT mainly underpredicted total and base flow in November 2009 at Palestine (Figure 2.4). Inability of the input rainfall data to completely reflect the actual spatial variability of rain, could lead to error in predicting flow (Santhi et al., 2001; Srinivasan et al., 1998).

Table 2.8. Parameters tweaked for calibration.

Variable	Input file	Recommended Values	Final Value
CN2	Management	35 – 98	±5%
ESCO	HRU	0 – 1	0.95
SOL_AWC	Soil	0 – 1	0.78-0.82
ALPHA_BF	Groundwater	0 – 1	0.77
RCHRG_DP	Groundwater	0 – 1	0.58
GWQMN	Groundwater	0 – 5000	0.2
GW_REVAP	Groundwater	0.02 – 0.2	0.03
CH_N2	Main Channel	0 – 0.3	0.08
USLE_P	Management	0 – 1	0.80
SPCON	Basin	0.0001 – 0.01	0.0016
SPEXP	Basin	1.0 –1.5	1.0
PRF	Basin	none	0.1
SURLAG	Basin	1 – 24	12
SOL_Z	Soil	0 – 3500	2032
CH_K2	Main channel	0 – 500	150
PHOSKD	Basin	100 – 200	175
PPERCO	Basin	10.0 – 17.5	11.0
CDN	Basin	0 – 3	3
SDNCO	Basin	0 – 1	0.2
NPERCO	Basin	0 – 1	0.32
RSDCO	Basin	0.02 – 0.1	0.02
SOL_NO3	Chemical	0 – 100	10 for all soil layers
SOL_ORGN	Chemical	0 – 100	10 for all soil layers
BIOMIX	Management	0 – 1	0.15
SHALLST_N	Groundwater	0 – 1000	5

Table 2.9. Results for the calibration and validation of the LRW SWAT model.

Gauge Site	Output	Calibration				Validation			
		R^2†	NSE‡	PBIAS§	RSR¶	R^2	NSE	PBIAS	RSR
Colt	Total flow	0.70	0.66	-4.28	0.58	0.78	0.68	7.68	0.48
	Surface flow	0.66	0.64	-1.16	0.60	0.80	0.64	-0.17	0.47
	Base flow			-3.81				21.11	
	Sediment	0.59	0.55	46.25	0.83	0.79	0.67	23.53	0.56
	Total P	0.58	0.52	18.25	0.89	0.84	0.72	28.30	0.44
	Nitrate N	0.53	0.50	4.47	0.96	0.69	0.54	0.11	0.65
Palestine	Total flow	0.58	0.54	-7.65	0.66	0.77	0.61	-13.46	0.61
	Surface flow	0.54	0.51	-10.94	0.73	0.80	0.72	-20.55	0.52
	Base flow			-12.57				-3.09	

†Coefficient of Determination
‡Nash-Sutcliffe Efficiency
§Percent Bias
¶Root mean square error-standard deviation ratio

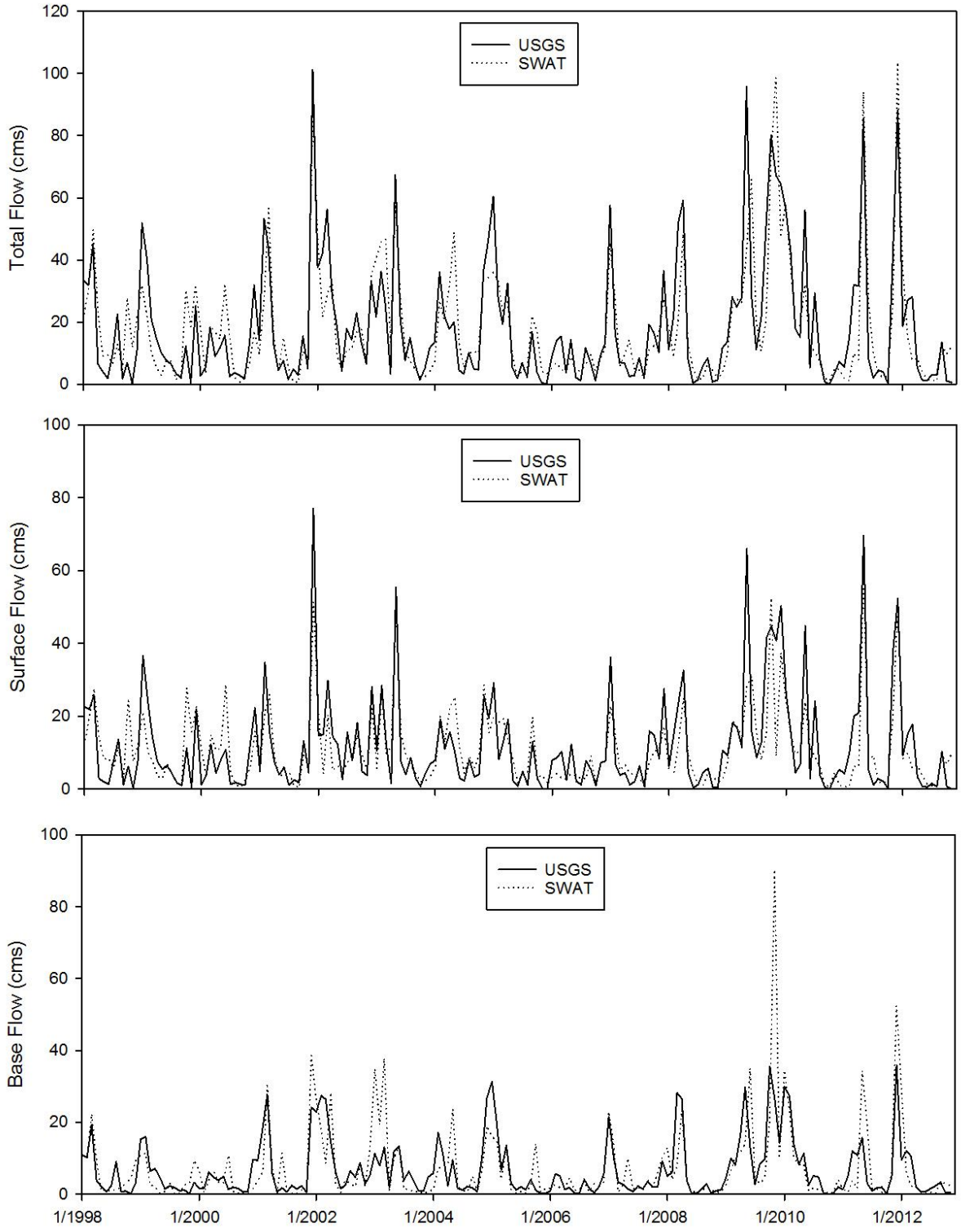

Figure 2.3. Graphical comparison of the measured and simulated flow data at the Colt site.

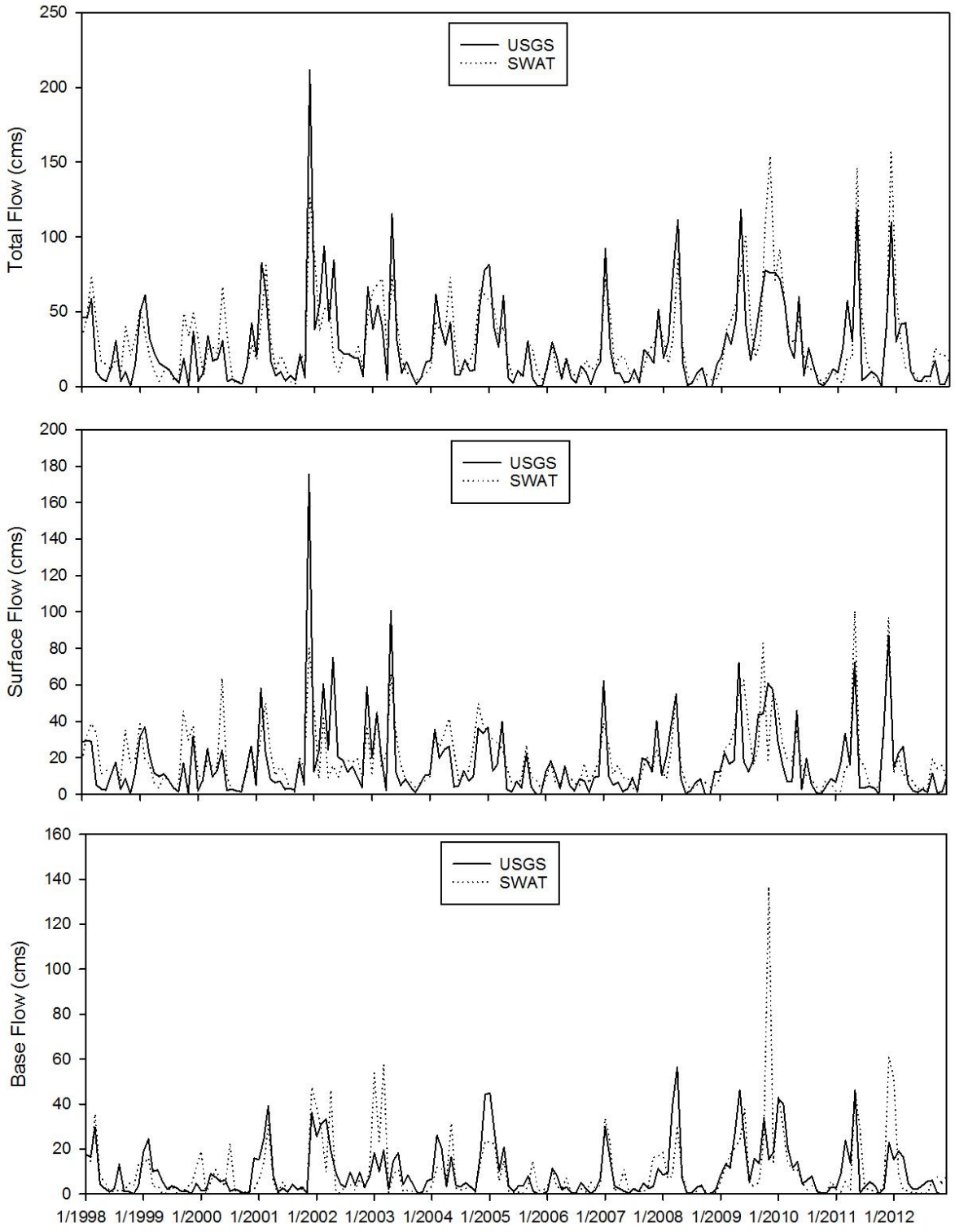

Figure 2.4. Graphical comparison of the measured and simulated flow data at the Palestine site.

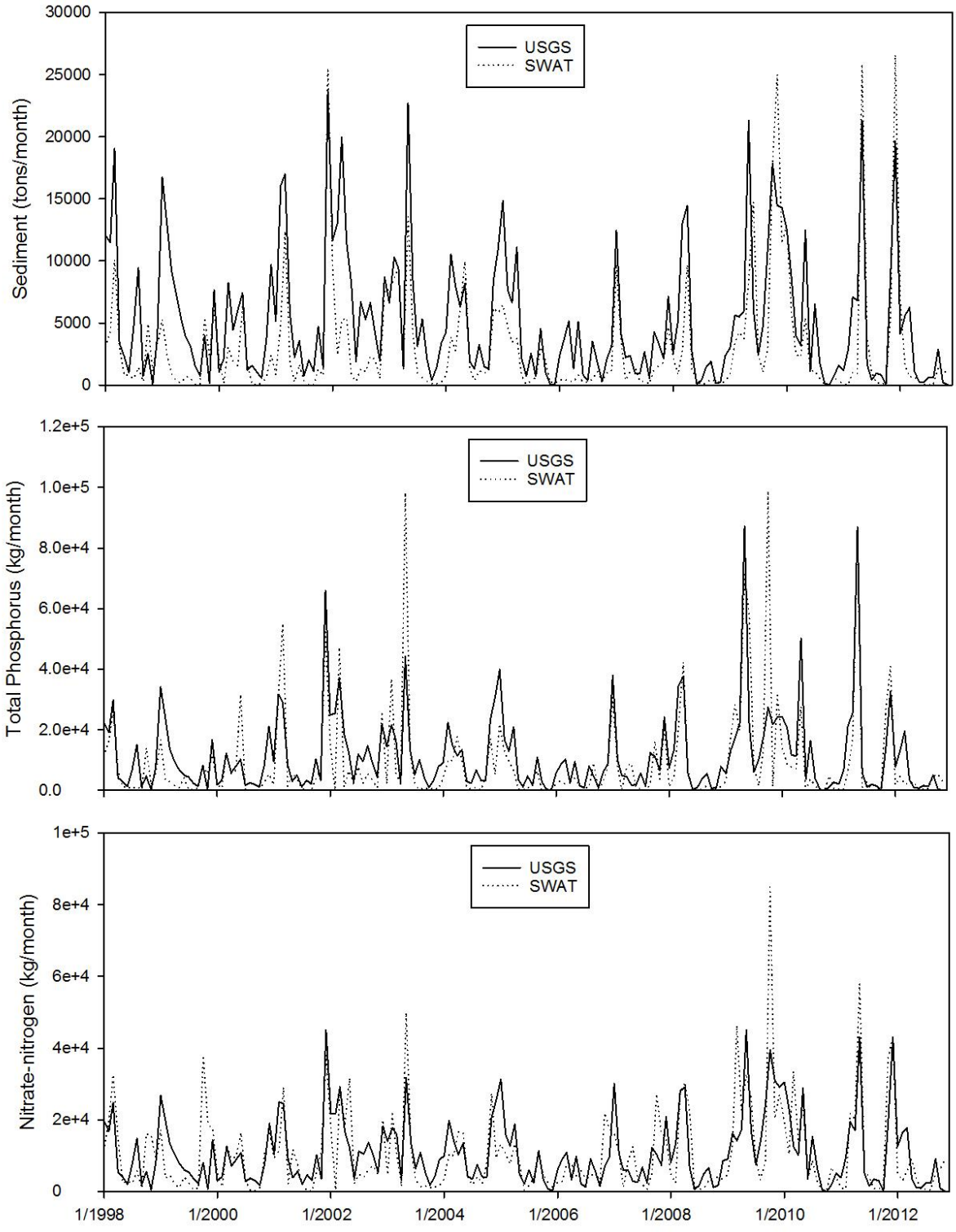

Figure 2.5. Graphical comparison of the measured and simulated water quality data at the Colt site.

Sediment was underpredicted by SWAT for the calibration and validation period at the Colt site. Sediment underprediction during March 2002 was likely to be propagated from flow underprediction which could be seen in Figure (2.5). As P binds with sediment and transports with it, TP was somewhat underpredicted for the calibration and validation period similar to sediment. Calibration and validation for NO3-N had some overpredicted peaks (October 1999 and November 2004) while there was underprediction in February 2002. In general, calibration of NO3-N is often difficult, resulting in poor simulations (Chu et al., 2004).

2.4.2 CP SCENARIOS

Individual CPs were simulated by SWAT. The simulated pollutant losses exiting the HRUs from each CP were compared with baseline loads, where no CPs were in place. The percentage load reductions associated with the selected CPs are shown in Figure (2.6).

In general, filter strips are used to reduce nutrient losses from upland areas of the watershed. It was the most effective CP in reducing predicted TN loss (40%), along with second most effective in reducing predicted TP loss (43%; Figure 2.6). Critical area planting was the most effective CP in reducing predicted sediment (80%) and predicted TP loss (58%), along with second most effective in reducing predicted TN (16%; Figure 2.6).

As grade-stabilization structures work to increase deposition in upstream channel reaches, these structures were most effective where channel degradation was the major source of sediments and nutrients discharge. Grade stabilization structures decreased predicted sediment loss (36%) to a relatively greater extent than predicted TP (5%) and predicted TN (2%; Figure 2.6). Irrigation land leveling resulted in a decrease in the HRU slope and slope length, which in turn influenced surface transport processes. As sediment and P are affected most by surface

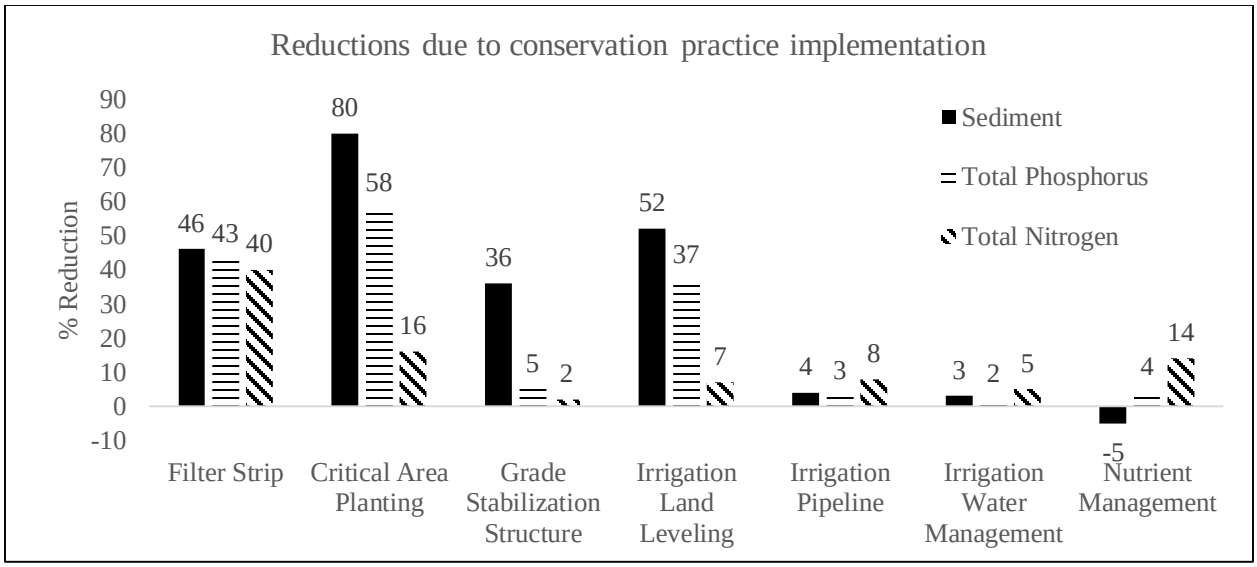

Figure 2.6. Percentage reductions in pollutant losses due to simulation of CPs.

processes, with N transport mainly a subsurface flow process, irrigation land leveling was more effective in reducing predicted sediment (52%; second most effective among CPs) and predicted TP (37%) loss as compared to predicted TN (7%) loss.

Irrigation pipeline, irrigation water management, and nutrient management were not predicted to be as effective at decreasing nutrient and sediment loss in LRW, as were the other selected CPs. Irrigation pipeline reduced sediment, TP, and TN by 5, 4, and 4%, respectively; and irrigation water management by 3, 2, and 2%, respectively. Nutrient management decreased TP and TN loss by 4 and 10%, respectively, but increased sediment loss slightly (5%; Figure 2.6) compared with baseline losses. This reduction in N and P losses with adoption of CP 590 (nutrient management), is consistent with the fact that this CP functions to decrease the rate of nutrients applied and at time when there is a lower risk of runoff or leaching. The slight increase in sediment loss compared to the baseline with nutrient management may have resulted from lower crop yields as a consequence of lower N and P applications (5%; Figure 2.6). The lower crop yields (overall 8% reduced yield) likely resulted in less vegetative cover and thus, more erodible exposed soil surfaces. This yield reduction was contrary to the expected impact of properly applied nutrient management that is intended to eliminate excess nutrient application, not to reduce yield (NRCS, 2013).

Overall effectiveness of CPs decreased in the order; critical area planting, filter strip, irrigation land leveling, grade stabilization structure, irrigation pipeline, irrigation water management, and nutrient management. Paired t-tests, performed to test differences between the predicted losses at each HRU, comparing baseline and CP datasets, revealed significant differences ($p < 0.05$) for all losses (i.e. sediment, TP, and TN). Moreover, paired t-tests between losses from each pair of CPs also revealed significant differences ($p < 0.05$).

Cumulative nutrient and sediment loss curves were developed at the HRU level to visualize the effectiveness of CPs (Figure 2.7). Only one CP, i.e., irrigation land leveling, is shown in Figure (2.7) for sediment, TP, and TN. White et al. (2009) presented several such curves to show pollutant losses variation with respect to the HRU area. The cumulative nutrient and sediment curves drawn in this study were sorted with respect to the contributing watershed area and arranged in ascending order with respect to the specific variable losses. This allowed the comparison of pre and post CP loss from the same HRU (Figure 2.7). It is evident that cumulative sediment, TP, and TN loss with CPs in place was less than that with no CPs for the same contributing watershed area. For instance, the post-CP sediment losses were approx. 35000 tons compared to the pre-CP sediment losses of approximately 75000 tons. As the contributing watershed area increased, the difference between with and without CP, widened more for sediment than TP and TN (Figure 2.7). The trend followed by TP was similar as that of sediment because of the ability of P to bind and transport with sediments. Moreover, sediment and TP are dominated by surface processes as opposed to TN that is dominated by underground processes. The reductions obtained were higher for sediments (52%) followed by TP (37%) and TN (7%). Kannan et al. (2014) also reported that the sediment reduction from irrigation land leveling CP (42%) was higher than that obtained by TN (35%).

2.5 CHALLENGES IN MODELING CPS

Although various CPs have been modeled with SWAT in this study, there are some weaknesses in simulating these CPs that need to be stated. While simulating filter strips, the filter strip placement and effectiveness throughout an HRU could vary and it would be difficult to uniformly represent all strips by the default parameters. Pasture was simulated on all barren

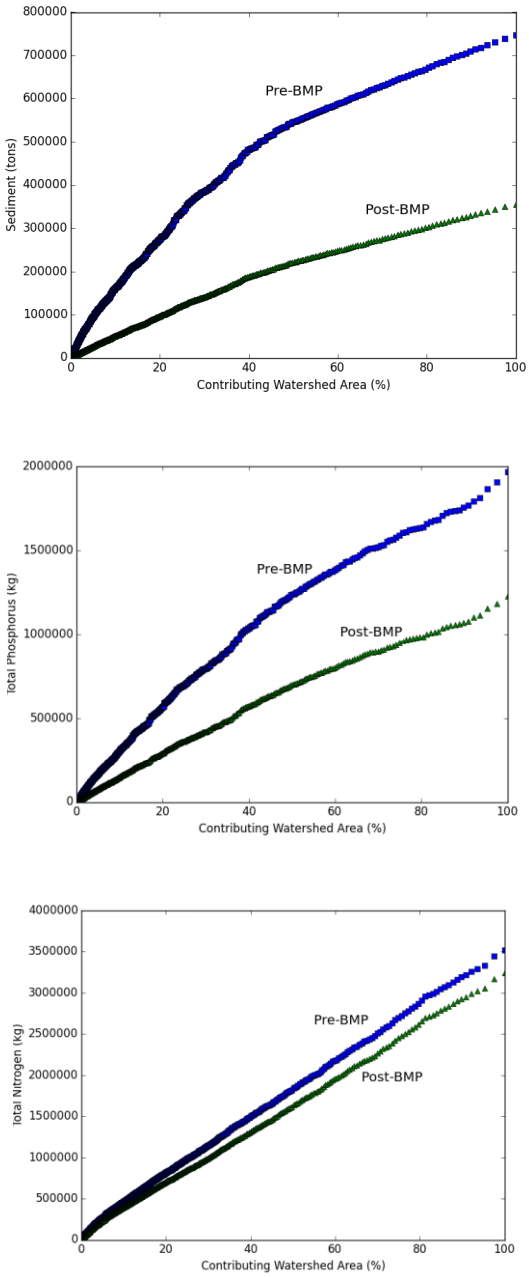

Figure 2.7. Cumulative sediment (top), total P (middle), and total N (bottom) losses by percent of contributing watershed area.

areas for defining critical area planting in this study. However, some barren areas such as rocky or eroded land might not be suitable for healthy pasture establishment.

Grade stabilization structures were represented in the model by reducing slope as 20% of the actual slope. However, some parts of a stream that do not have high slopes, might not be candidates for this CP. Irrigation land leveling, which was represented in the model by reducing slope and slope length, might not be useful in areas with an existing irrigation system where there is no need of irrigation land leveling. Moreover, the same irrigation land leveling methodogies might not be applied to different crop conditions. Irrigation pipeline, which was uniformly represented in the model by improving conveyance efficiency, may not be applicable to all systems in the HRU. At present, a user has no ability to place the pipeline at a specific location in the HRU where they may be needed.

Irrigation water management controls the amount of irrigation water for efficient utilization of available water. Controlling application of irrigation water should not only be based on rainfall occurring on a specific day, but also should depend on the amount of antecedent rainfall. Modeling nutrient management CP was a failure in our study. We reduced applied fertilizers by half to represent a nutrient management scheme. However, the results depict an increase in sediment losses and decrease in crop yield, although TP and TN losses were reduced. Nutrient management should not recommend that the applied fertilizers be reduced uniformly regardless of soil nutrient status and crop requirements. Proper simulation of this CP would require knowledge (and detailed input data) of the field specific soil and crop conditions. There might be a possibility that nutrient management programs have already been implemented in some fields.

2.5 SUMMARY

The effect of implementing MRBI-recommended CPs in LRW were estimated with SWAT. The model setup procedure was enhanced by incorporating wetlands in CAST land-use layers and row crops in NLCD land-use layers. Moreover, no thresholds were used for land use, soil, and slope, to minimize loss of spatial information, better represent hydrological processes, and accurately assess water quality results. The model was setup with the most recent available datasets, calibrated and validated for total flow, surface flow, base flow, sediment, TP, and NO3-N at the Colt site and for total flow, surface flow, and base flow at the Palestine site.

The statistical results for the calibration and validation were found to be satisfactory or better except a few RSR values for the calibration period. The critical area planting CP was the most effective practice in reducing overall predicted nutrient and sediment loss followed by filter strip, irrigation land leveling, grade stabilization structure, irrigation pipeline, irrigation water management, and nutrient management. In the future, additional work is needed to model CPs accurately in the watershed, requiring detailed knowledge of the range of practices already existing. Representing the mechanisms of the CP accurately in SWAT is a requirement to insure that simulation results provide appropriate guidance to users.

2.6 ACKNOWLEDGEMENTS

The authors would like to thank various state and federal agencies for providing the much-needed input data. Thanks also go to the staff members Arkansas High Performance Computing Center (AHPCC) at the University of Arkansas for helping with the resources needed to compile SWAT on supercomputer. The funding for this project was provided by the Arkansas Natural Resources Commission (ANRC - Project 11-1900 FY 11 CWA Section 319(h)) and Natural Resources Conservation Services (NRCS Agreement No. 68-7103-10-393).

2.7 REFERENCES

Alexander, R. B., Smith, R. A., Schwarz, G. E., Boyer, E. W., Nolan, J. V. & Brakebill, J. W. (2008). Differences in phosphorus and nitrogen delivery to the Gulf of Mexico from the Mississippi river basin. *Environ. Sci. & Tech., 42*(3), 822-830.

ANRC. (2012). State of Arkansas Nutrient Reduction Strategy. Little Rock, Ark.: ANRC. Available at: https://static.ark.org/eeuploads/anrc/AR_Nutrient_Reduction_Strategy_101014.pdf. Accessed 8 December 2015.

Arabi, M., Govindaraju, R. S., Hantush, M. M., & Engel, B. A. (2006). Role of watershed subdivision on modeling: The Effectiveness of Best Management Practices with SWAT. *J. American Water Res. Assoc*., *42*(2), 513-528.

Center for Advanced Spatial Technologies - CAST. (2007). Land use land cover: Fall 2006 (raster). Center for Advanced Spatial Technologies, Fayetteville, AR. Available at: http://www.geostor.arkansas.gov. Accessed 8 December 2015.

Chiang, L., Chaubey, I., Gitau, M. W., & Arnold, J. G. (2010). Differentiating impacts of land use changes from pasture management in a CEAP watershed using the SWAT model. *Trans. ASABE, 53*(5), 1569-1584.

Chu, T., Shirmohammadi, A., Montas, H., & Sadeghi, A. (2004). Evaluation of the SWAT model sediment and nutrient components in the piedmont physiographic region of Maryland. *Trans. ASABE, 47*(5), 1523-1538.

Engel, B., Storm, D., White, M., Arnold, J., & Arabi, M. (2007). A hydrologic/water quality model application protocol. *J. Am. Water Works Assoc., 43*(5), 1223-1236.

Espinoza, L., Slaton, N. A. & Mozaffari, M. (2006). The soil test report. Little Rock, AR: University of Arkansas Division of Agriculture. Available at: http://www.uark.edu/depts/soiltest/NewSoilTest/pdf_files/FSA-2153.pdf. Accessed 8 December 2015.

Espinoza, L., Slaton, N. A. & Mozaffari, M. (2007). Understanding the number on your soil test report. Little Rock, AR: University of Arkansas Division of Agriculture. Available at: http://www.uark.edu/depts/soiltest/NewSoilTest/pdf_files/FSA-2118.pdf. Accessed 8 December 2015.

Folle, S., Dalzell, B., & Mulla, D. (2007). Evaluation of best management practices (BMPs) in impaired watersheds using the SWAT model. St. Paul, MN: Minnesota Department of Agriculture.

Gitau, M. (2003). A quantitative assessment of BMP effectiveness for phosphorus pollution control: The Town Brook Watershed, New York. Ph.D. Dissertation. The Pennsylvania State University, Pennsylvania, USA, 247 pp.

Gorham, B., & Tullis, J. (2007). Final Report: 2006 Arkansas Land Use and Land Cover (LULC). Fayetteville, Ark.: Arkansas Natural Resource Commission.

Hall, L., Christensen, T., Bramblett, J., Hopkins, S., & Hubbert, J. (2012). USDA's National Water Quality Initiative: A Watershed Academy Webcast, July 2012. Presented on the U.S. Environmental Protection Agency's Watershed Academy Webcast. Available at http://water.epa.gov/learn/ training/wacademy/upload/2012_07_10-slides.pdf. Accessed 8 December 2015.

Her, Y., Frankenberger, J., Chaubey, I., & Srinivasan, R. (2015). Threshold effects in HRU definition of the Soil and Water Assessment Tool. *Trans. ASABE*, 58(2), 367-378.

Homer, C., Huang, C., Yang, L., Wylie, B., & Coan, M. (2004). Development of a 2001 national land-cover database for the United States. *Photogrammetric Eng. and Remote Sensing, 70*(7), 829-840.

Kalin, L., & Hantush, M. M. (2006). Hydrologic modeling of an eastern Pennsylvania watershed with NEXRAD and rain gauge data. *J. Hydrol. Eng., 11*(6), 555-569.

Kannan, N., Jeong, J., & Srinivasan, R. (2011). Hydrologic modeling of a canal-irrigated agricultural watershed with irrigation best management practices: Case study. *J. Hydrologic Eng., 16*(9), 746-757.

Kannan, N., Omani, N., & Miranda, R. (2014). Water quality modeling of an agricultural watershed with Best Management Practices. *International J. of Res. in Eng. Tech., 3*(1), 553-564.

Maringanti, C. (2008). Development of a multi-objective optimization tool for the selection and placement of BMPs in a watershed for NPS pollution control. M.S. thesis. Purdue, Ind.: Purdue University, Department of Agricultural and Biological Engineering.

Martinez-Martinez, E., Nejadhashemi, A. P., Woznicki, S. A., & Love, B. J. (2014). Modeling the hydrological significance of wetland restoration scenarios. *J. Environ. Mgt., 133*, 121-134.

Meals, D. W., & Hopkins, R. B. (2002). Phosphorus reductions following riparian restoration in two agricultural watersheds in Vermont, USA. *Water Sci. Tech., 45*, 51–60.

Mehlich, A. (1984). Mehlich 3 soil test extractant: A modification of Mehlich 2 extractant. *Commun. Soil Sci. Plant Anal., 15*, 1409-1416.

Moon, J., Srinivasan, R., & Jacobs, J. H. (2004). Streamflow estimation using spatially distributed rainfall in the Trinity River basin, Texas. *Trans. ASABE, 47*(5), 1445-1451.

Moriasi, D., Arnold, J., Van Liew, M., Bingner, R., Harmel, R., & Veith, T. (2007). Model evaluation guidelines for systematic quantification of accuracy in watershed simulations. *Trans ASABE, 50*(3), 885-900.

Neitsch, S. L., Arnold, J. G., Kiniry, J. G., & Williams, J. R. (2011). Soil and Water Assessment Tool: Theoretical Documentation. Version 2009. Technical Report No. 406. College Station, TX: Texas Water Resources Institute.

NRCS. (2013). Site-specific nutrient management: for nutrient management planning to improve crop production, environmental quality, and economic return. Available at http://www.agronext.iastate.edu/soilfertility/nutrienttopics/4r/Site-SpecificNutrientManagementPlanning_ver2.pdf. Accessed 10 December 2015.

Pai, N., and D. Saraswat. 2011. SWAT2009_LUC: A tool to activate the land use change module in SWAT 2009. *Trans. Am. Soc. Agric. Biol. Eng.* 54(5): 1649-1658.

Pai, N., Saraswat, D., & Daniels, M. (2011). Identifying priority subwatersheds in the Illinois River Drainage Area in Arkansas watershed using a distributed modeling approach. *Trans. ASABE, 54*(6), 2181-2196.

Perez, M., & Walker, S. (2014). "Improving water quality: A review of the Mississippi River Basin Healthy Watersheds Initiative (MRBI) to target U.S. farm conservation funds." Working Paper. Washington, DC: World Resources Institute. Available online at: wri.org/ publication/MRBI. Accessed 8 December 2015.

Runkel, R. L., Crawford, C. G., & Cohn, T. A. (2004). Load Estimator (LOADEST): A FORTRAN program for estimating constituent loads in streams and rivers. Reston, Va.: US Department of the Interior, US Geological Survey. Available at: http://pubs.usgs.gov/tm/2005/tm4A5/pdf/508final.pdf. Accessed 8 December 2015.

Santhi, C., Arnold, J. G., Williams, J. R., Dugas, W. A., Srinivasan, R., & Hauck, L. M. (2001). Validation of the SWAT model on a large river basin with point and nonpoint sources. *J. Am. Water Res. Assoc., 37*(5), 1169-1188.

Santhi, C., Srinivasan, R., Arnold, J. G., & Williams, J. (2006). A modeling approach to evaluate the impacts of water quality management plans implemented in a watershed in Texas. *Environ. Modelling & Software, 21*(8), 1141-1157.

Sharpley, A. N., Jones, C. A., Gray, C., & Cole, C. V. (1984). A simplified soil and plant phosphorus model: II. Prediction of labile, organic, and sorbed phosphorus. *Soil Sci. Soc. Am. J., 48*, 805-809.

Sharpley, A., Jarvie, H. P., Buda, A., May, L., Spears, B., & Kleinman, P. (2013). Phosphorus legacy: Overcoming the effects of past management practices to mitigate future water quality impairment. *J. Environ. Qual., 42*(5), 1308-1326.

Srinivasan, R., Ramanarayanan, T., Arnold, J., & Bednarz, S. (1998). Large area hydrologic modeling and assessment part II: model development. *J. American Water Res. Assoc., 34*(1), 91-101.

U.S. Department of Agriculture Mississippi River Basin Healthy Watersheds Initiative Web site. (2015a). Available at

http://www.nrcs.usda.gov/wps/portal/nrcs/detail/ia/programs/landscape/?cid=nrcs142p2_007958. Accessed 8 December 2015.

U.S. Department of Agriculture Mississippi River Basin Healthy Watersheds Initiative Web site. (2015b). Available at http://www.nrcs.usda.gov/wps/portal/nrcs/detailfull/national/home/?cid=stelprdb1048200. Accessed 8 December 2015.

U.S. Department of Agriculture - Natural Resources Conservation Service. (2009). Mississippi River Basin Healthy Watershed Initiative. Available at http://www.nrcs.usda.gov/Internet/FSE_DOCUMENTS//nrcs143_008142.pdf Accessed 8 December 2015.

USDA - Natural Resources Conservation Service. (2014). Mississippi River Basin Healthy Watersheds Initiative – MRBI. U.S. Govt. Printing Office, Washington, DC. 6 pages. Available at: http://discoveryfarms.uark.edu/MBRI_Fact_Sheet.pdf. Accessed 8 December 2015.

U.S. Environmental Protection Agency. (1996). Environmental indicators of water quality in the United States. EPA 841-R-96-002. USEPA, Office of Water (4503F), U.S. Govt. Printing Office, Washington, DC.

U.S. Environmental Protection Agency. (2003). National management measures to control nonpoint source pollution from agriculture. (EPA 841-B-03-004)

Vadas, P. A., & White, M. J. (2010). Validating soil phosphorus routines in the SWAT model. *Trans. ASABE. 53*(5):1469-1476.

White, M. J., Harmel, R. D., Arnold, J. G., & Williams, J. R. (2014). SWAT check: A screening tool to assist users in the identification of potential model application problems. *J. Environ. Qual., 43*(1), 208-214.

White, M. J., Storm, D. E., Busteed, P. R., Stoodley, S. H., & Phillips, S. H. (2009). Evaluating nonpoint source critical source area contributions at the watershed scale. *J. Environ. Qual., 38*(4), 1654-1663.

3. CP TARGETING TOOL: A TOOL TO AID IN CP PLAN DEVELOPMENT

3.1 ABSTRACT

Studies have indicated that nutrients are one of the main causes of NPS impairment of surface waters in the United States. Use of CPs is advocated to control the generation and delivery of nutrients to water bodies. Watershed models can serve as a tool to estimate the long-term impacts of CPs on water quality. Considerable expertise in modeling and spatial sciences is needed to identify suitable CPs and locations for placement. In this research, a tool was developed that allows integrating user-defined targeted areas utilizing either a single or multiple selection criteria with the SWAT model. The tool has been designed to simulate CPs at the lowest simulation level, an HRU level, of the SWAT model by building a new targeting procedure for SWAT applications and decision-making. It is a Python-based tool that uses open source packages such as GDAL and Matplotlib. There are seven major components of the tool: SWAT project folder, target area, processing, run pre-CP SWAT model, define CP, run post-CP SWAT model, and results display. In addition to the integration of already-developed SWAT models, this tool also resolves the non-spatial nature of HRUs issue with SWAT. The tool could help modelers and watershed planners make decisions for CP placement.

3.2 INTRODUCTION

CPs often play an important role in minimizing generation and delivery of nutrients from agricultural sources (U.S. Environmental Protection Agency, 2003). Monitoring long-term effectiveness of CPs can be time consuming. As a result, hydrologic and water quality models are used as an alternative for assessing CP impacts on water quality (Arabi et al., 2006).

Although published data for general CP effectiveness values are available, models can generate reliable estimates of various CP implementation strategies on nutrient loss in a given watershed.

One of the most widely used and accepted hydrologic, water quality model – SWAT – has been extensively used to identify target areas contributing relatively higher sediment, and nutrient loads to receiving waters (Niraula et al., 2012; Panagopoulos et al., 2011; Pai et al., 2011). Although the SWAT model has been widely used in identifying target areas, the tool is complex for watershed planners to use. Watershed planners need tools to accurately predict CP effectiveness without expending time dealing with the technical details of model complexities. Thus, a user-friendly tool is needed that can run complex models like SWAT in the background, to accurately assess CP affects, in terms of sediment and nutrient reductions on targeted areas. However, there are two challenges that need to be addressed before such a tool can be developed.

1. Non-spatial nature of HRUs

Non-spatial nature of HRUs implies that HRUs are discontinuous landmasses in a subwatershed (Gassman et al., 2007; Pai et al., 2012). The SWAT simulation occurs at the HRU level that includes uniformly applied management operations and CP impacts on hydrology and water quality. As HRU forms the basic unit of spatial optimization in the SWAT model, accurate simulation of CPs on target HRUs must be done. A CP's effectiveness is highly site-specific (Gitau et al., 2004). As a result, associating HRUs to specific spatial locations is an important step before proceeding with the simulation of CP effects. If the HRUs are discontinuous and lumped together units, it is not possible to accurately simulate CPs on specific spatial locations in the model. Identification of HRUs that matches the target area of interest is a challenging task, as the non-spatial nature issue of HRUs needs to be handled. Identical HRUs in a subwatershed have the same HRU IDs. If a user wants to simulate a CP in one part of the subwatershed where

the target HRU is, SWAT will simulate the same CP in other parts of the subwatershed where HRUs with same IDs are present. As a result, new targeting procedures should be built for SWAT applications to simulate CPs on correct spatial locations or subset areas of an HRU.

2. Already-developed SWAT models

Building a new SWAT model, and calibrating or parameterizing it for a specific watershed is a rigorous and time-consuming process (Hormann et al., 2009). The targeting tool can be equipped with a capability to use the already-developed SWAT models. Usage of already-developed models in the tool will have a dual benefit watershed planner will not be required to go through the complexity of SWAT model, and usability of SWAT model will be increased among the modeling community. Singh and Saraswat (2016) developed a simulation approach and compared it with the conventional SWAT modeler's approach for targeting biofuel crop production on marginal lands in LRW, Arkansas. Although Singh and Saraswat (2016) developed a new approach to simulate biofuel crops only on targeted HRUs, the approach could not be used with already-developed (calibrated and validated) SWAT models. In other words, the new simulation approach could only be used while setting up a new SWAT model. As a result, usage of already-developed models in the tool requires building new targeting procedures for SWAT applications and decision-making.

There have been attempts in the past to develop tools to assess water quality impacts due to land use changes or CP implementations. For example, the AGWA tool allows targeting of areas by drawing polygons on a visible watershed map in its interface. The AGWA tool was developed to integrate landscape information with process models (SWAT or kinematic runoff and erosion model) to assess watershed impacts. The LTHIA model was developed by the Agricultural and Biological Engineering Department at Purdue University for estimating changes

in runoff, recharge, and NPS pollution resulting from land use changes (LTHIA, 2013). The tool ingests land use and soil patterns and predicts average annual runoff and NPS loads for specific land use patterns. A web-based software tool has been developed for optimizing CPs in watersheds (Babbar-Sebens et al, 2015). The tool integrates with SWAT; however, the non-spatial nature of the HRUs was not addressed in the tool.

Houston Engineering, Inc. developed an Ag BMP Assessment and Tracking Tool in collaboration with Heron Lake Watershed District to disseminate information on the use and effectiveness of agricultural CPs in the state of Minnesota (ABATT, 2014). The Ag BMP Assessment and Tracking Tool includes a comprehensive database of information related to agricultural CPs in Minnesota, a web-based CP assessment tool, and CP tracking tool. Tetra Tech developed the SET tool to design sites and evaluate pollutant-loading rates per requirements of Pulaski County, Arkansas (SET, 2000). The tool provides a framework to allow the performance standards or conservation design approaches for evaluating and testing site designs. The Virginia Tech BMP decision support software assists in the selection of CPs for the treatment of stormwater runoff (EPA, 2008). Any of the available CP classes (ponds and basin, infiltration, filtration, wetland, manufactured/proprietary CP, and LID) and their criteria (contributing drainage area, impervious fraction of CPs contributing drainage area, soils, geologic site constraints, other CP implementation considerations, and performance goals) can be defined in the tool. However, a lack of spatial target components or inability to simulate CPs makes the above tools ineffective for assessing CP impacts on a user-defined target area with correct spatial definition.

The objective of this study was to assess the effectiveness of CPs on a user-defined target area using a CP tool, a graphical user interface developed for SWAT for targeting CPs. The tool

was also evaluated for the CRW. The tool could help in assessing CP impacts prior to their implementation on target areas of interest.

3.3 PACKAGES FOR TOOL DEVELOPMENT

Python language (version 3.4) was used for coding the tool (Python Software Foundation, 2015). Python is a high-level programming language and allows programmers to express concepts in fewer lines of code. It is developed under an Open Source Initiative (OSI) - approved open source license that is distributed by Python Software Foundation. The packages required for setting up the CP tool are shown in Appendix A1. Descriptions of some of the major packages that were installed are as follows:

- GDAL is an open source translator library for raster and vector geospatial data formats (OSGeo, 2015). GDAL supports numerous raster and vector data formats such as GeoTIFF, ESRI Shapefile, and GML. These data formats can be translated and processed with the help of command line utilities of GDAL.
- Matplotlib library produces publication quality figures in a variety of formats (http://matplotlib.org/index.html). Matplotlib.pyplot, which is a collection of command style functions, is used to create a figure, decorates the plot with labels, etc.
- Numpy package is used for scientific computing with Python and has a powerful N-dimensional array object (http://www.numpy.org/). It is also used as an efficient multi-dimensional container of generic data where arbitrary data types can be defined.
- PyQt, developed by Riverbank Computing Limited, is a Python binding for the QT cross-platform framework (https://wiki.python.org/moin/PyQt).

- SciPy is a collection of open source software for scientific computing with a specific core package such as Python, NumPy, Matplotlib, etc. (http://www.scipy.org/).

3.4 DESCRIPTION OF VARIOUS COMPONENTS IN THE CP TOOL

The layout of the CP tool is shown in Figure (3.1). The flowchart depicting the front-end processes for the user is shown in Figure (3.2). Approximately 900 lines of code was written in Python to build the tool (Appendix A2). The interface for the CP tool was developed with Python PyQt package for interface creation. Basically when a user clicks a button or menu option, the object emits a signal. The connect() method is used to connect the signal with a slot, in this case the slot is a Python callable method. For example, the user clicks the Input SWAT Project Folder menu item and it in turn emits a signal which is then connected to the Python method responsible for prompting the user to select their SWAT folder. Various components in the CP tool are as follows:

Input SWAT Project Folder: The SWAT projects that has already been developed and calibrated/validated can be used as an input by clicking on the SWAT Project Folder button under the Input menu.

Input Targeted Area: Target area can be defined by the user with a single criterion (Strijker, 2005) or multiple criteria (Gopalakrishnan et al., 2011). The CP tool allows the input of target area in the form of a binary raster layer by clicking on the Targeted Area button under the input menu. The binary raster layer should have values 1 and 0 as targeted and non-targeted area, respectively.

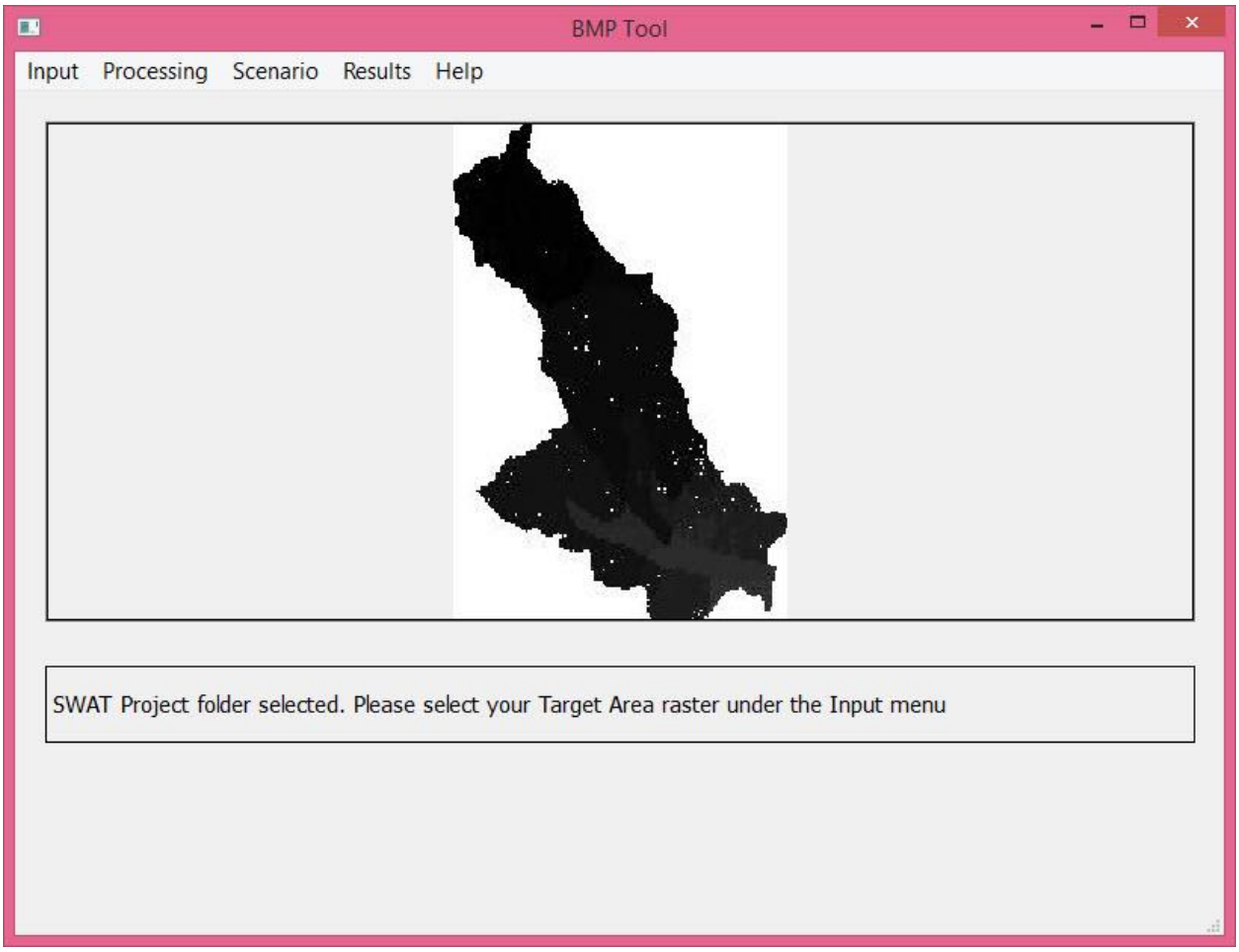

Figure 3.1. Layout of the CP tool showing the main menu, the watershed image in the image viewer, and an area for status messages and guidance for the user.

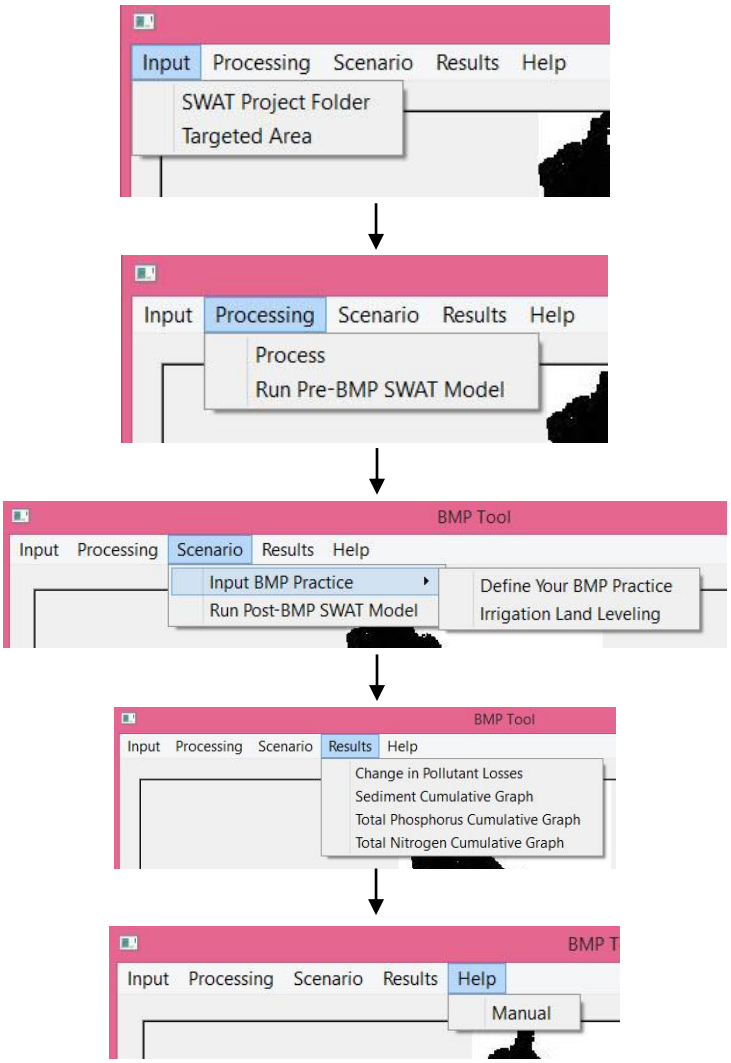

Figure 3.2. Flowchart depicting the front-end processes in the CP tool.

Process: HRUs are generally delineated by defining thresholds for land use, soil, and slope (Neitsch et al. 2011). The target HRUs were identified by processing HRU raster (obtained from the input SWAT project folder) and target area layer with the Process button available under the Processing menu.

Run Pre-CP SWAT Model: Once the Run SWAT model button under the Processing menu is clicked, the SWAT model was run in the background of the tool for the baseline scenario. The baseline results for sediment, TP, and TN losses are simulated and stored in this step.

Input Your CP Practice: CPs are simulated on the target area in this step. The tool provides an option to input a CP practice by clicking on the Input Your CP Practice button under the Scenario menu. This can be done by modifying the relevant parameters in the SWAT project folder's text files that represents the CP of interest. A simple CP, namely, irrigation land leveling was predefined in the tool that could be simulated on target area with the help of a mouse click.

Run Post-CP SWAT Model: The SWAT model was run again in the background of the tool for the CP scenario by clicking on the Run SWAT Model button under the Scenario menu. The CP results for sediment, TP, and TN losses are simulated and stored in this step.

Results: The baseline and CP results for sediment, TP, and TN losses will be compared in this step. The changes in pollutant losses obtained after the CP simulation can be viewed by clicking on the Changes in Pollutant Losses button. Cumulative pollutant losses against the contributing watershed area can be viewed for sediment, TP, and TN variables by clicking on Sediment, TP, or TN button under the Results menu. This will give an idea of the variation in trends of cumulative pollutant losses between the baseline and CP scenario.

Help: There is also a help menu that links to this document in case the user has any challenges in understanding the tool.

3.5 User Involvement and Background Processes for Each Step in the CP Tool.

The background processes are much more involved and complex as compared to the front-end processes. The user interacts with the seven major steps in the tool. Each step performed by the user results in activation of background processes for the respective step. The flowchart depicting the back-end processes is shown in Figure (3.3).

The user steps and what happens in the background is described below:

Step 1: Identify SWAT project folder

What happens in the background: In addition to identifying the location of the SWAT project folder, the hrus1 raster also gets located and get converted to a JPG file so that the watershed's HRUs can be displayed in the image viewer of the tool. gdal_translate was used to convert hrus1 from GRID to JPG. Pillow (aka PIL)'s Image was used to open the JPG and PyQt's QPixmap was used to draw it in the image viewer.

Step 2: Identify targeted area

What happens in the background: Again gdal_translate, PIL, and QPixmap was used to render the target raster in the image viewer of the CP tool. The binary method assumes you already have a binary raster that can be uploaded.

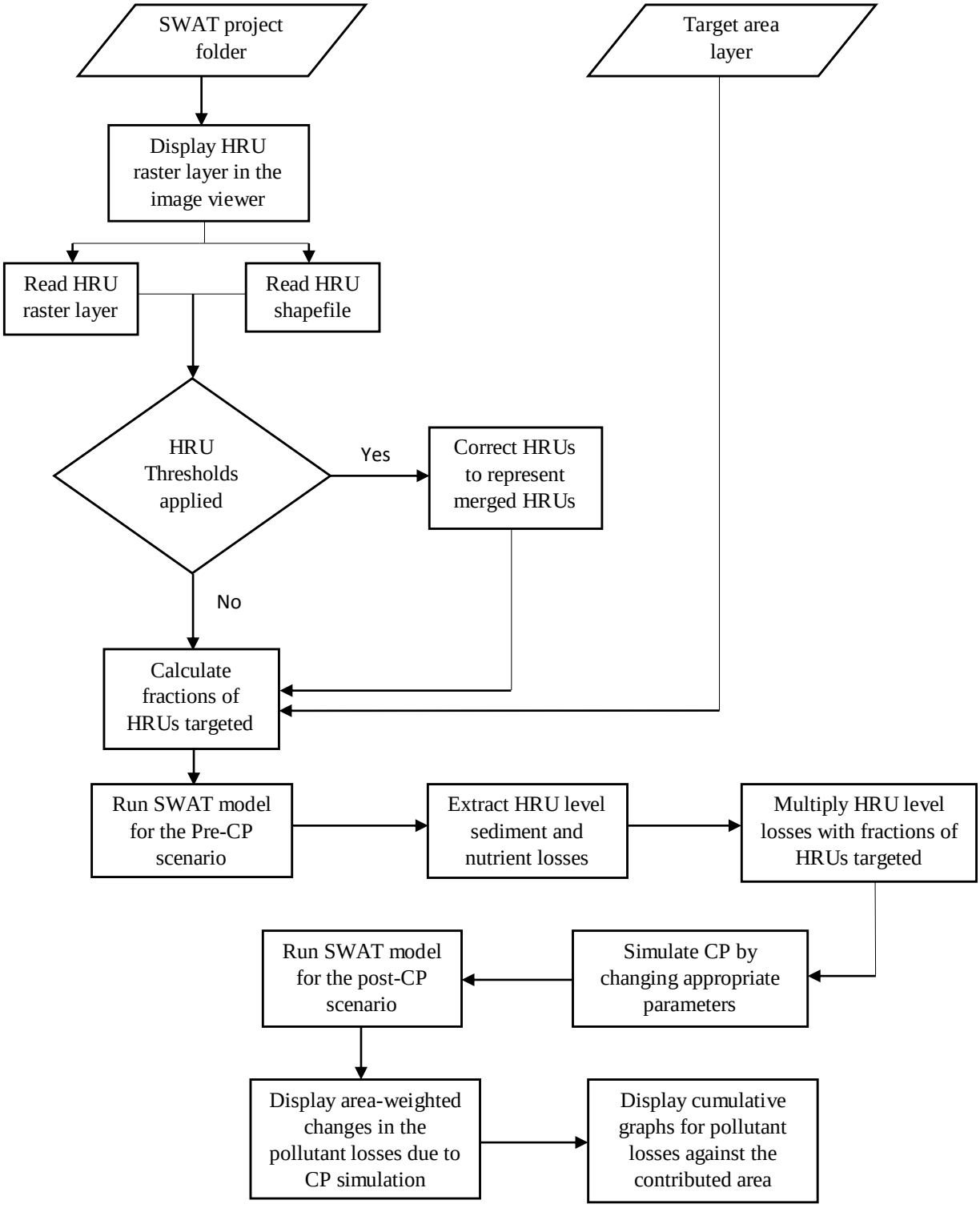

Figure 3.3. Flowchart depicting the back-end processes in the CP tool.

Step 3: Process

What happens in the background: Once the target HRUs have been identified, the targeted component for each HRU was calculated with the help of process button. The targeted component ranges from 0 to 1 with 0 being no targeting component and 1 being all targeting component for that HRU. For example, suppose there are three spatially discontinuous similar HRU areas represented by HRU ID 1 in a subwatershed. Two out of three HRUs overlap with the binary target area. This will result in a targeting component of 2/3=0.667. As can be seen in Figure (3.4), the first matrix represents the targeted area matrix with T being the targeted area and 0 being the non-targeted area, the second matrix represents the pre-processed HRU matrix with 8 HRU IDs, and the third matrix represents the post-processed HRU matrix or targeted HRU matrix with target area HRUs accompanied by letter 't'. Numpy was used to find the index locations where the targeted area is located, in order to flag those cells with 't'.

If threshold percentages have been applied for land use, soil, and slope, in the SWAT model, a post-threshold HRU raster layer will be developed by the tool as no such layer is available in the SWAT project folder. For developing a post-threshold raster layer that can be used as a pre-processed HRU matrix, the HRU raster and HRU shape file was used from the input SWAT project folder (Pai and Saraswat, 2011). The Euclidean distance approach was followed to create a layer that represents post-threshold HRUs. Once the post-threshold layer has been used, the target HRU IDs and their targeting components can be identified.

T	0	0	T
T	T	0	T
T	0	T	0
T	0	0	T

1	1	2	2
1	8	3	7
4	5	6	6
4	8	7	2

1T	1	2	2T
1T	8T	3	7T
4T	5	6T	6
4T	8	7	2T

Figure 3.4. Identification of targeted HRUs represented in a matrix form: (i) Target area matrix, (ii) Pre-processed HRU matrix, and (iii) Post-processed HRU matrix.

Step 4: Run Pre-CP SWAT model

What happens in the background: The header row was extracted in the output.hru file and area was split using area.split(.) command. The HRU area values get extracted in this step. Respective to the area, the pollutant loss values get extracted. The extracted area and pollutant loss information is also based on the crops defined at this point. Specific SWAT codes for the crops should be entered in the appropriate crop column in the Python code. Sometimes, the index position for the columns in the output.hru file changes. This problem was tackled by placing appropriate if statements in the codes.

Step 5: Input CP

What happens in the background: Irrigation land leveling CP was defined in the tool. The glob and regex modules were used to locate the HRU files and find specific places in the files where the substitution for relevant parameters need to be performed. Specifically, HRU slope (reducing HRU_SLP parameter by 10%) and slope length (reducing SLSUBBSIN parameter by one-tenth of the default value) were modified (Kannan et al., 2011). Alternatively, a user can navigate in the SWAT project folder and modify relevant parameter for the CP of interest.

Step 6: Run Post-CP SWAT model

What happens in the background: Same as step 4.

Step 7: Results

What happens in the background: Matplotlib was used for creating and putting legends on the plots. The accumu() function was used to calculate the percentage area and accu() function was used to calculate total respective pollutant losses for the pre- and post- CP scenario. The

pollutant losses were extracted corresponding to the HRU area and the area-weighted pollutant losses (sum-product of pollutant losses and area divided by sum of area) were generated. The area-weighted pollutant averages for the pre- and post- CP scenario are calculated in this step.

3.6 EVALUATION PROCEDURE

The tool was tested for CRW, Arkansas (hydrologic unit code – 08020302). The drainage area for CRW is 5066 square kilometers and cover portions of 11 counties: Clay, Craighead, Cross, Greene, Jackson, Lawrence, Monroe, Poinsett, Prairie, Randolph, and Woodruff. Land uses and land covers in the CRW are shown in Table (3.1). A random area (only row crops) in the CRW was selected as the target area to simulate irrigation land leveling – a predefined CP in the tool.

The cumulative graphs were generated for the three water quality variables: sediment, TP, and TN. In these graphs, the trend of water quality variables can be analyzed over contributing area before and after the simulation of irrigation land leveling. The cumulative graph for sediment variation over contributing area in the CRW is shown in Figure (3.5). The cumulative graph was generated by the CP tool with the help of matplot package coded in the background of the tool. It can be clearly seen that the cumulative sediment coming out of contributing watershed area for the post-CP scenario shows a decreasing trend as compared to the pre-CP scenario. Similar trends for the TP and TN variation over contributing area in the CRW can be seen in Figures (3.6) and (3.7).

Table 3.1: Land use and land covers in Cache River Watershed.

Land Cover	% of watershed area
Soybean	39
Forest	25
Rice	14
Corn	9
Cotton	3
Pasture	3.5
Urban	2.8
Barren	2.1
Water	1.6

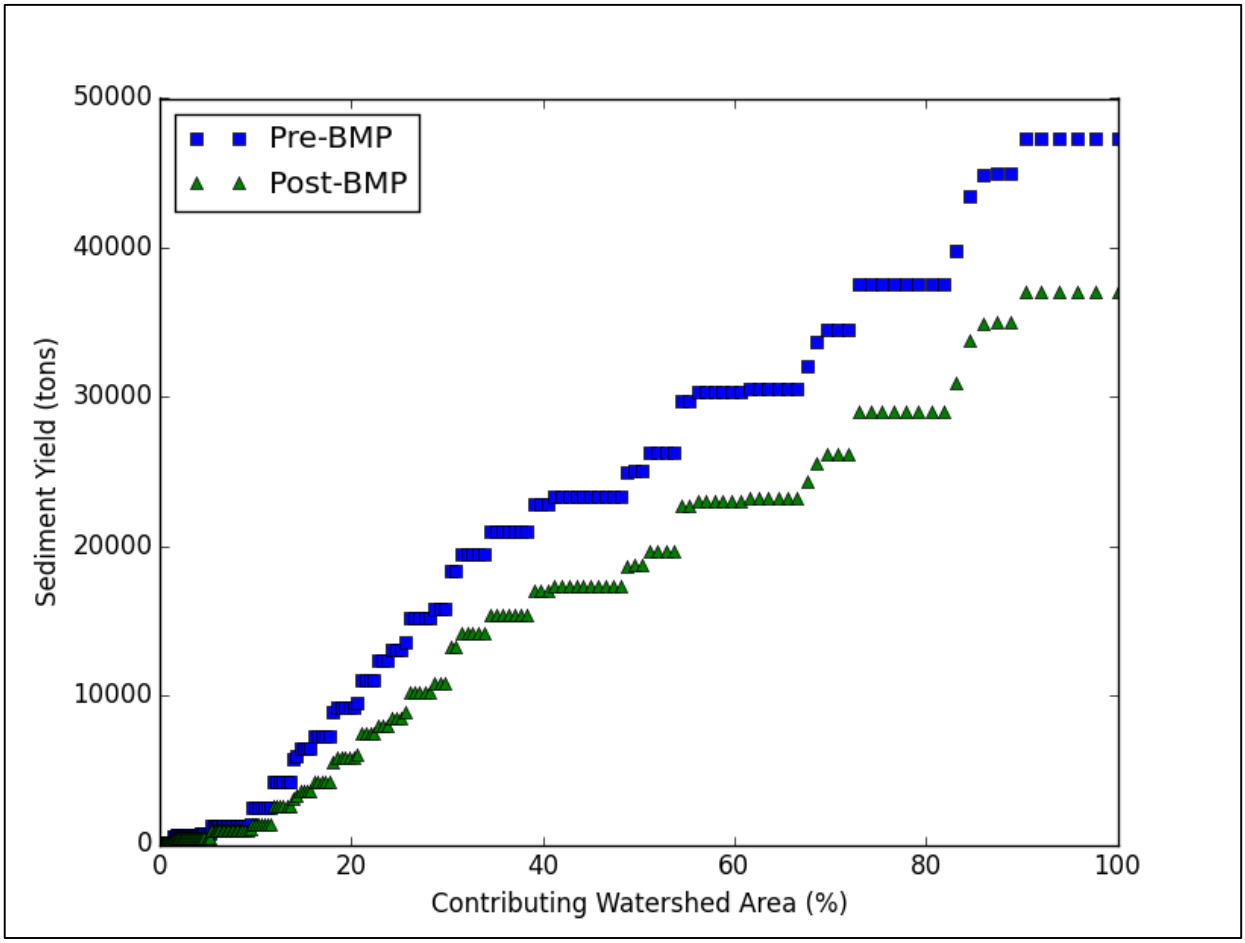

Figure 3.5. The cumulative graph for sediment variation over contributing area before and after simulation of the irrigation land leveling CP in a targeted area of the Cache River watershed.

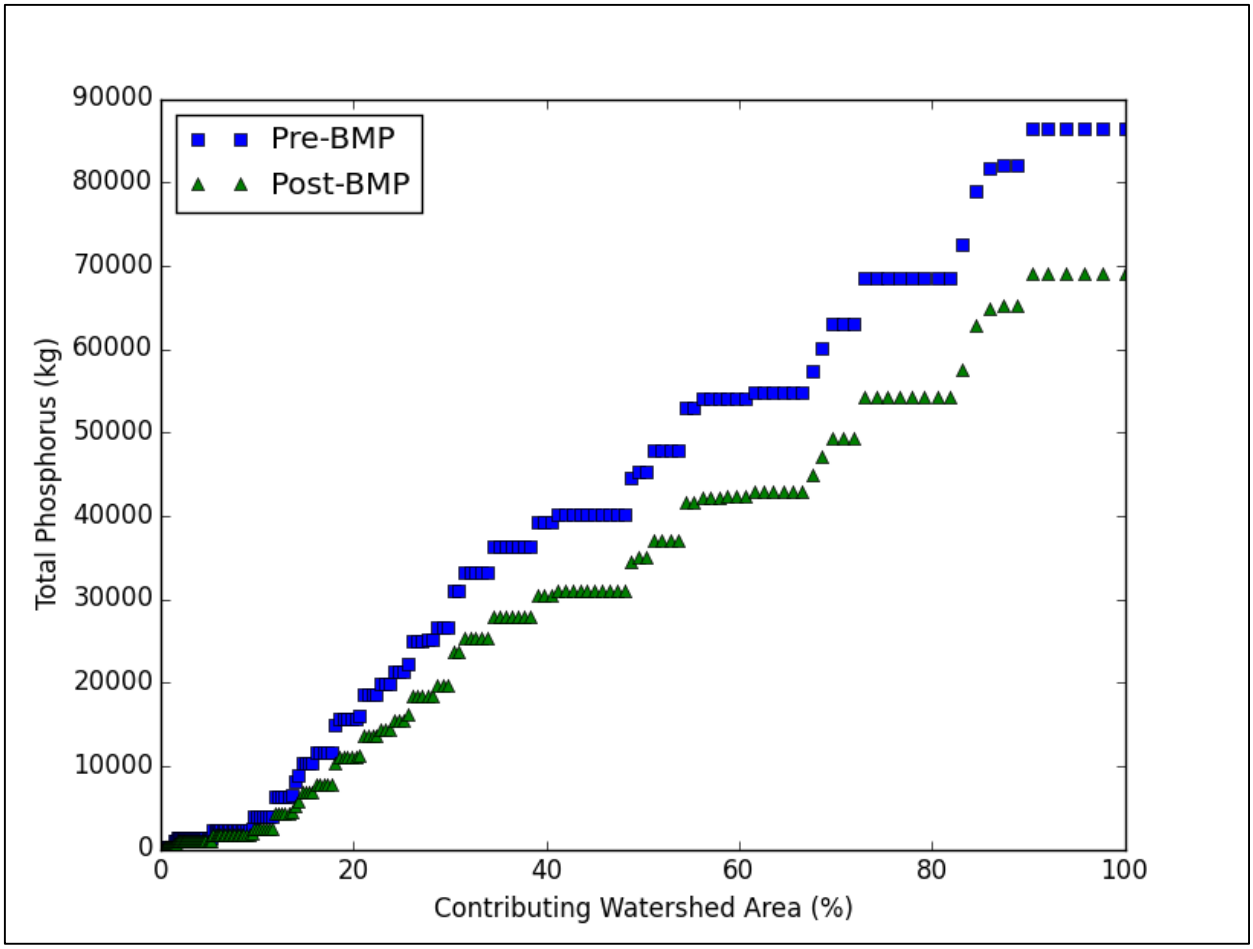

Figure 3.6. The cumulative graph for total phosphorus variation over contributing area before and after simulation of the irrigation land leveling CP in a targeted area of the Cache River watershed.

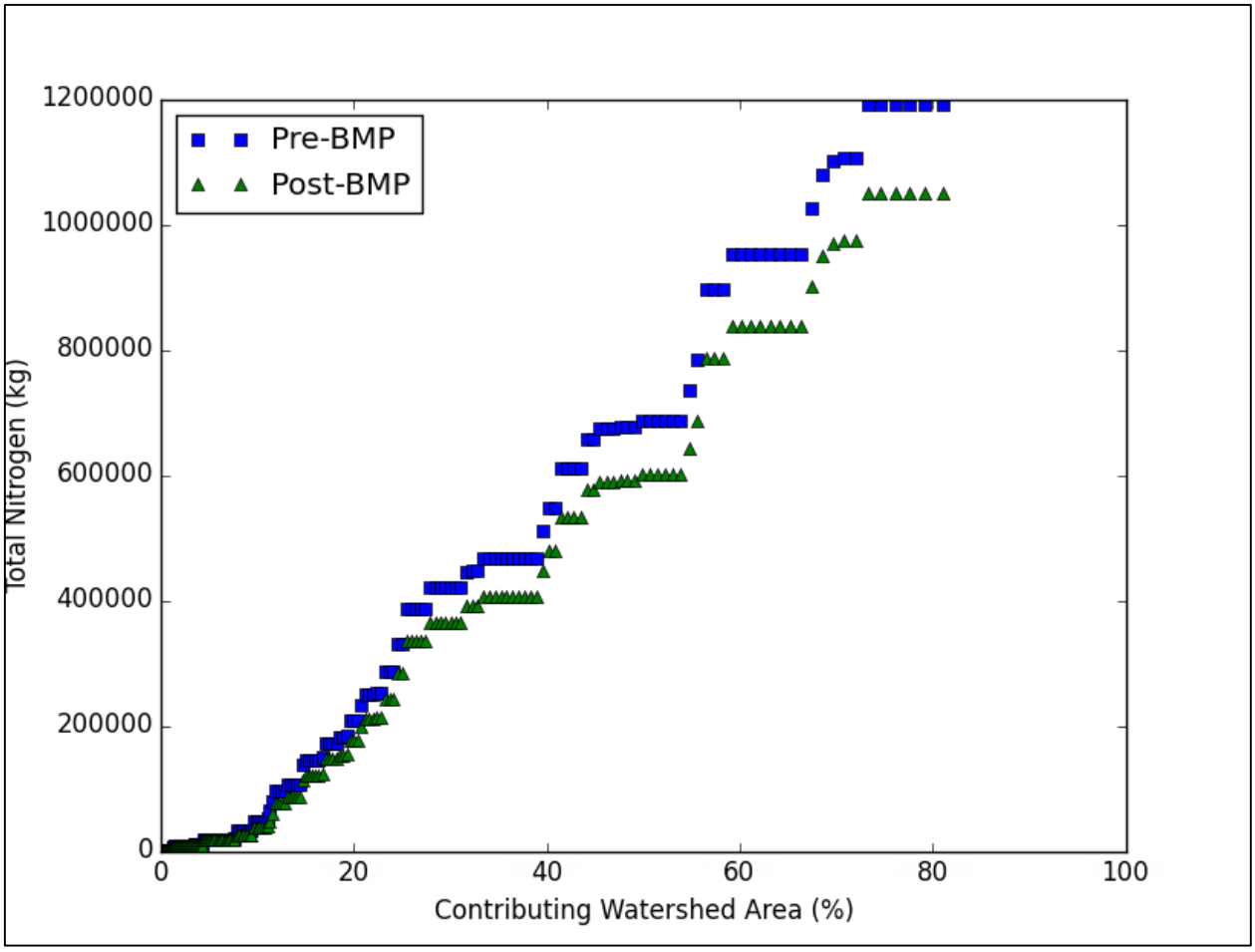

Figure 3.7. The cumulative graph for total nitrogen variation over contributing area before and after simulation of the irrigation land leveling CP in a targeted area of the Cache River watershed.

The percentage reduction in pollutant loseses due to simulation of irrigation land leveling CP can be seen in Figure (3.8). The area-weighted average sediment, total phosphorus, and total nitrogen losses for the pre and post-CP scenario is shown in table (3.2). There was a 22% percentage decrease in sediment losses, 20% decrease in TP losses, and 12% decrease in TN losses. The area-weighted average pollutant losses were less for the post-CP scenario compared to pre-CP scenario.

The CP selected in this validation, irrigation land leveling, was modeled very simply by reducing the HRU slope by 10% and slope length by one-tenth of the default value. This was used to illustrate the method of application of the tool. In practice, each CP should be selected for simulation with caution. CP mechanisms should be accurately reflected by the change in model parameters. The target area should be defined by the user and input in the tool. The user should possess data on the actual field conditions relating to the CP in the area. This will ensure that the user simulates CPs on targeted areas where they are needed and where implementation will be adequately simulated.

There is a need to further develop the tool with a capability to simulate a set of fields with varying soil and crop conditions that would allow input of multiple areas and specification by the user of unique input data and management practices for each.

3.7 SUMMARY

An open source-code Python-based CP tool was developed to simulate CPs on user-defined target areas. The tool uses PyQt software for interface creation as well as open source automate the process for (i) targeting areas at the lowest simulation level, (ii) simulating CP on

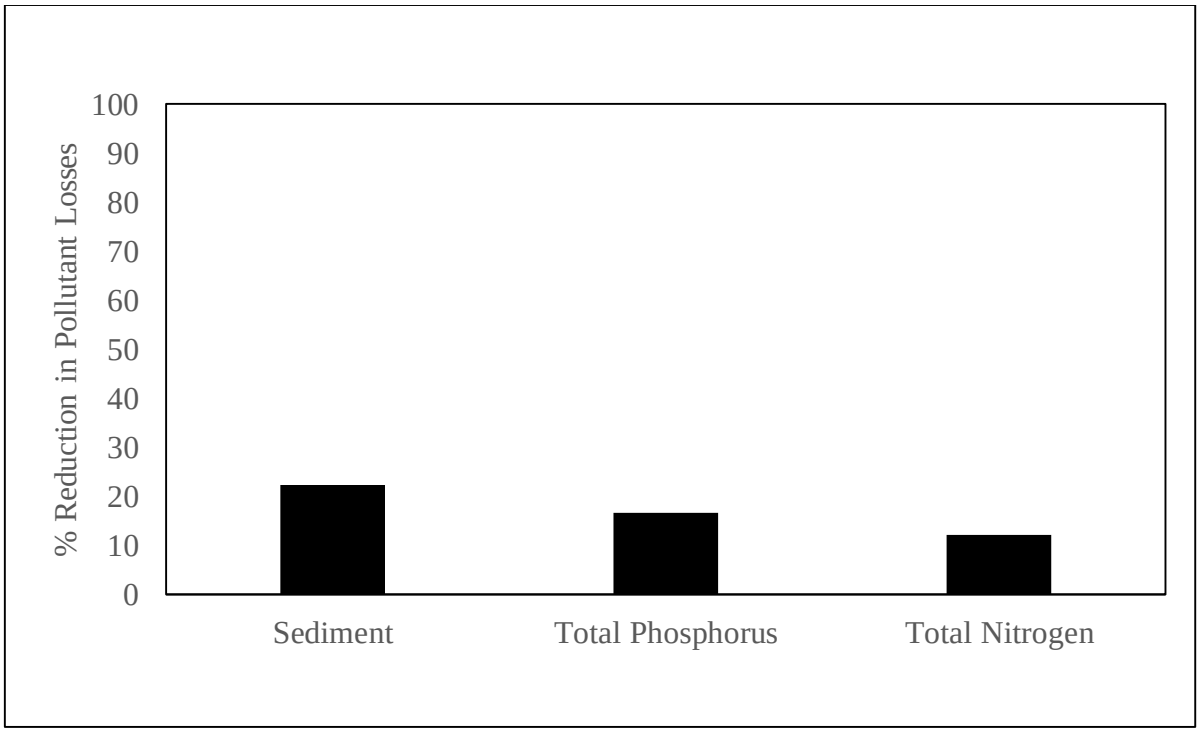

Figure 3.8. Percentage reduction in sediment, TP, and TN losses due to the simulation of the irrigation land leveling CP in the targeted area.

Table 3.2: Area-weighted average sediment, total phosphorus, and total nitrogen losses for the pre and post-CP scenario.

Output Variable	Pre-CP	Post-CP
Area-weighted average sediment yield (tons)	0.09	0.07
Area-weighted average total phosphorus (kg)	0.06	0.05
Area-weighted average total nitrogen (kg)	0.58	0.51

packages such as GDAL and MatplotLib. The tool is desktop-based, standalone, and user-friendly. The tool inputs the already-developed SWAT models and target area, and the target area, and (iii) visualizing results and analyzing differences between the baseline and CP scenarios. The tool was tested for the CRW and a reduction of 22% in sediment losses, 20% in TP losses, and 12% in TN losses was found. The tool provides a quick approach to provide an insight to the watershed planners and modelers to address the water quality impacts on a specific target area.

3.7 REFERENCES

ABATT. (2014). Ag CP Assessment and Tracking Tool. Fargo, N.D..: Houston Engineering, Inc. Available at http://agcp.houstoneng.net/. Accessed 26 October 2015.

Arabi, M., Govindaraju, R. S., Hantush, M. M., & Engel, B. A. (2006). Role of watershed subdivision on modeling: The Effectiveness of Best Management Practices with SWAT. *J. American Water Res. Assoc*., *42*(2), 513-528.

Babbar-Sebens, M., Mukhopadhyay, S., Singh, V. B., and Piemonti, A. D. (2015). A web-based software tool for participatory optimization of CPs in watersheds. *Environ. Modelling & Software, 69*, 111-127.

EPA. (2008). Gulf hypoxia action plan 2008 for reducing, mitigating and controlling hypoxia in the northern Gulf of Mexico and improving water quality in the Mississippi River Basin.

Gassman, P. W., Reyes, M. R., Green, C. H., & Arnold, J. G. (2007). The Soil and Water Assessment Tool: Historical development, applications, and future research directions. *Trans. ASABE, 50*(4), 1211-1250.

Gitau, M.W., Veith, T.L., & Gburek, W.J. (2004). Farm-level optimization of CP placement for cost-effective pollution reduction. *Trans. ASABE, 47*, 1923-1931.

Gopalakrishnan, G., Cristina Negri, M., & Snyder, S. W. (2011). A Novel Framework to Classify Marginal Land for Sustainable Biomass Feedstock Production. *J. Environ. Qual., 40*(5), 1593-1600.

Hormann, G., Koplin, N., Cai, Q., & Fohrer, N. (2009). Using a simple model as a tool to parameterise the SWAT model of the Xiangxi river in China. *Quaternary International, 208*(1), 116-120.

Kannan, N., Jeong, J. & Srinivasan, R. (2011). Hydrologic modeling of a canal-irrigated agricultural watershed with irrigation best management practices: Case study. *J. Hydrologic Eng., 16*(9), 746-757.

LTHIA. (2013). Long Term Impact Hydrological Assessment. West Lafayette, In.: Purdue University. Available at https://engineering.purdue.edu/~lthia/. Accessed 26 October 2015.

OSGeo. (2015). GDAL: Geospatial Data Abstraction Library. Available at http://www.gdal.org/. Accessed 30 November 2015.

Neitsch, S. L., Arnold, J. G., Kiniry, J. G., & Williams, J. R. (2011). Soil and Water Assessment Tool: Theoretical Documentation. Version 2009. Technical Report No. 406. College Station, TX: Texas Water Resources Institute.

Niraula, R., Kalin, L., Wang, R., & Srivastava, P. (2012). Determining nutrient and sediment critical source areas with SWAT: effect of lumped calibration. *Trans. ASABE*, 55(1), 137-147.

Pai, N., & Saraswat, D. (2011). SWAT2009_LUC: A tool to activate the land use change module in SWAT 2009. *Trans. ASABE, 54*(5), 1649-1658.

Pai, N., Saraswat, D., & Daniels, M. (2011). Identifying priority subwatersheds in the Illinois River Drainage Area in Arkansas watershed using a distributed modeling approach. *Trans. ASABE, 54*(6), 2181-2196.

Pai, N., Saraswat, D., & Srinivasan, R. (2012). Field_SWAT: A tool for mapping SWAT output to field boundaries. *Computers and Geosci., 40*, 175-184.

Panagopoulos, Y., Makropoulos, C., Baltas, E., & Mimikou, M. (2011). SWAT parameterization for the identification of critical diffuse pollution source areas under data limitations. *Ecological Modelling, 222*(19), 3500-3512.

Python Software Foundation. (2015). Python Language Reference, version 3.4. Available at http://www.python.org. Accessed 22 November 2015.

SET. (2000). Site evaluation tool user guide and documentation for the Lake Maumelle drainage basin, Pulaski County, Arkansas. Pasadena, Ca.: Tetra Tech, Inc. Available at: http://co.pulaski.ar.us/pdf/SETUserGuidanceV14.pdf. Accessed 26 October 2015.

Singh, G., & Saraswat, D. (2016). Development and evaluation of targeted marginal land mapping approach in SWAT model for simulating water quality impacts of selected second generation biofeedstock. *Environ. Modelling & Software, In Press*, DOI: 10.1016/j.envsoft.2015.12.001.

Strijker, D. (2005). Marginal lands in Europe--causes of decline. *Basic Appl. Ecol*., *6*(2), 99-106.

U.S. Environmental Protection Agency. (2003). National management measures to control nonpoint source pollution from agriculture. (EPA 841-B-03-004).

4. DEVELOPMENT OF A SWAT COMPATIBLE DESKTOP-BASED INTERACTIVE TARGETING TOOL TO SIMULATE CPS ON TARGET AREAS

4.1 ABSTRACT

Every year, billions of dollars are spent by taxpayers in the USA to alleviate water quality related concerns. CPs are typically used for improving water quality. Watershed models are frequently relied on to predict long-term impacts of CPs on water quality. Many studies have used the SWAT model to identify locations for CP implementation. However, identifying suitable locations for CP placement requires considerable experience in spatial sciences and modeling. Usage of already-developed SWAT models for assessing CP effectiveness is another area where research is required. As a result, a Python-based tool has been developed in this study that uses past-developed SWAT models and simulates CPs at the user-defined locations using an interactive CP simulation approach. The tool allows the user to select target area with mouse clicks in a user-friendly and interactive environment. The tool has been tested for LRW located in northeastern Arkansas. A target area was selected interactively and filter strip and irrigation land leveling was simulated. Predicted losses were decreased by 70%, 68%, and 47%, respectively, for sediment, TP, and TN.

4.2 INTRODUCTION

Models are useful in simulating CPs in targeted areas. Identifying and targeting critical areas with high pollution potential is a more efficient way for allocating resources and controlling NPS pollution than random or uniform application of a strategy across a wider area (Diebel et al., 2008; White et al., 2009).

Models such as SWAT can be used for water quality assessment at the regional (Demissie et al., 2012), watershed (Sarkar et al., 2011), subwatershed (Pai et al., 2011), and even at the smallest simulation scale or HRU (Pai et al., 2012). The SWAT model has also been extensively used to identify areas contributing relatively higher sediment and nutrients to the receiving water body (Niraula et al., 2012; Panagopoulos et al., 2011; Pai et al., 2011). Without modeling studies, the water quality impacts of CPs are difficult to be appreciated at the regional, watershed, or even at the subwatershed level within a limited time frame.

NPS pollution models have many benefits for assessing effectiveness of CPs and selecting appropriate type and location for CPs, but is difficult for users who have little knowledge about model complexities. As a result, simplified tools have been developed for models such as pasture P management (PPM) and Texas CP Evaluation Tool (TBET). These tools might help watershed planners to assess effectiveness of CPs without expending time dealing with model complexities. For the SWAT modeling community, if such simplified tools facilitate integration of the already-developed SWAT models, then that will also result in the increase of already-developed SWAT models among the community.

A Python-based desktop binary version of the CP tool has already been developed as a part of Objective 2 of the dissertation. The CP tool runs SWAT in the background to accurately assess CP impacts in terms of sediment and nutrient reductions on targeted areas. The CP tool also resolves issues relating with the non-spatial nature of HRUs and integration of the already-developed SWAT models within the tool. However, the simplest way of selecting a target area of interest is to define a polygon (with mouse clicks) to mark the project area interactively (Parmenter, 2007). Making the CP tool interactive would be an advancement to the CP targeting tools. Therefore, the interactivity would allow the user to select the target area interactively and

simulate CPs on the selected target area.

The overall objective of this study was to accurately simulate CPs on target area with the help of CP tool. Specifically, the tool allows the user to interactively select their target area of interest and simulate CPs for assessing its impacts on sediment, TP, and TN.

4.3 TOOL COMPONENTS

Approximately 950 lines of code was written in Python to build the tool (Appendix A3). The components of the CP tool are the same as listed for the Chapter 3 of the dissertation: SWAT project folder, target area, process, run pre-CP SWAT model, defining CP practice, run post-CP SWAT model, and results. However, for the user-defined target area, the user has the option to select the target area interactively on the watershed map visible in the image viewer. The interactive method collects user-clicks to generate a binary raster that can be uploaded. The non-interactive method explained in Chapter 3 assumes you already have a binary raster that can be uploaded. The general process of going from user-clicks to a binary raster is:

- Converting pixel coordinates to geographic coordinates,
- Creating a shapefile mask,
- Using shapefile mask to create a raster mask with its values set to 1,
- Creating a new raster with the same extent as the original HRUs raster, but with its values set to 0,
- Using the raster mask to convert the new raster cells from 0 to 1 where the mask overlays the new raster, and
- Finally using that new raster (binary raster) as the uploaded file (target area).

4.4 EVALUATION PROCEDURE

The tool was tested for LRW, Arkansas. The LRW is an agricultural dominated watershed located in Mississippi Delta ecoregion of east central Arkansas and is designated by the hydrological unit code (HUC) 08020205 (Seaber, 1994). The drainage area for the LRW is 2,474 square kilometers and covers a portion of Craighead, Cross, Lee, Poinsett, St. Francis, and Woodruff counties. The LRW is a relatively flat watershed (90 percent of area has slope from 0 to 3 percent). Land cover in LRW are shown in Table (4.1). The NRCS reported LRW as a priority watershed for nutrients under the 2011-2016 NPS Pollution Management Plan (ANRC, 2012).

When using the tool, once the SWAT project was selected, the watershed map becomes visible in the image viewer. A target area of 70 km^2 comprising of row crops in Cross County was interactively-selected on the watershed map with the help of mouse clicks and a polygon was generated. The tool then reclassifies the selected target area to a binary layer.

The binary target layer was processed with the HRU layer to generate targeted and non-targeted component of HRUs. The simulated CPs were filter strip and irrigation land leveling. The filter strip was simulated in SWAT using the default parameters for the in-built filter strip module. The parameters include ratio of field area to filter strip area (FILTER_RATIO = 40), fraction of the HRU which drains to the most concentrated ten percent of the filter strip area (FILTER_CON = 0.5), and fraction of the flow within the most concentrated ten percent of the filter strip which is fully channelized (FILTER_CH = 0). The irrigation land leveling was simulated by reducing the HRU slope (HRU_SLP parameter) by 10% and slope length (SLSUBBSN) by one-tenth of the default value (Kannan et al., 2011). Both the CPs were simulated simultaneously on the croplands of the interactively-selected target area. The

Table 4.1: Land Cover in L'Anguille River Watershed (CAST, 2007).

Land Cover	% of watershed area
Corn	4.5
Cotton	6.9
Rice	14.9
Soybean	43.6
Specialty Crops	1.2

cumulative graphs were generated for sediment, TP, and TN.

The cumulative graph for sediment variation over contributing area in the LRW is shown in Figure (4.1). The graph shows that the cumulative sediment coming out of contributing watershed area for the post CP scenario has a decreasing trend as compared to the pre CP scenario. Similar trends can be seen for the TP and TN variation over contributing area in the LRW in Figures (4.2) and (4.3).

The percent reduction in pollutant losses due to the simulation of filter strip and irrigation land leveling CP can be seen in Figure (4.4). There was a 70% percentage decrease in sediment losses, 68% decrease in TP losses, and 47% decrease in TN losses. White and Arnold (2009) reviewed 22 published studies relating with filter strips and reported that the effectiveness of a filter strip can vary from 24-100% for sediment, 21-100% for TP, and 23-98% for TN. The pollutant reductions achieved in this study fall within the ranges listed by White and Arnold (2009). The area-weighted average pollutant losses for the pre and post-CP scenario can be seen in Table (4.2).

It should be noted that before defining the target area interactively, the user should be aware of the watershed and target area conditions. This will ensure that the right target area can be input in the tool and a relevant CP can be assessed for its water quality impacts. The tool itself does not have the capability to recommend a specific CP for the input target area. The user should define both the target area and the CP. Furthermore, care must be used in defining the SWAT parameters or management practice changes tht accurately reflect any simulated CP.

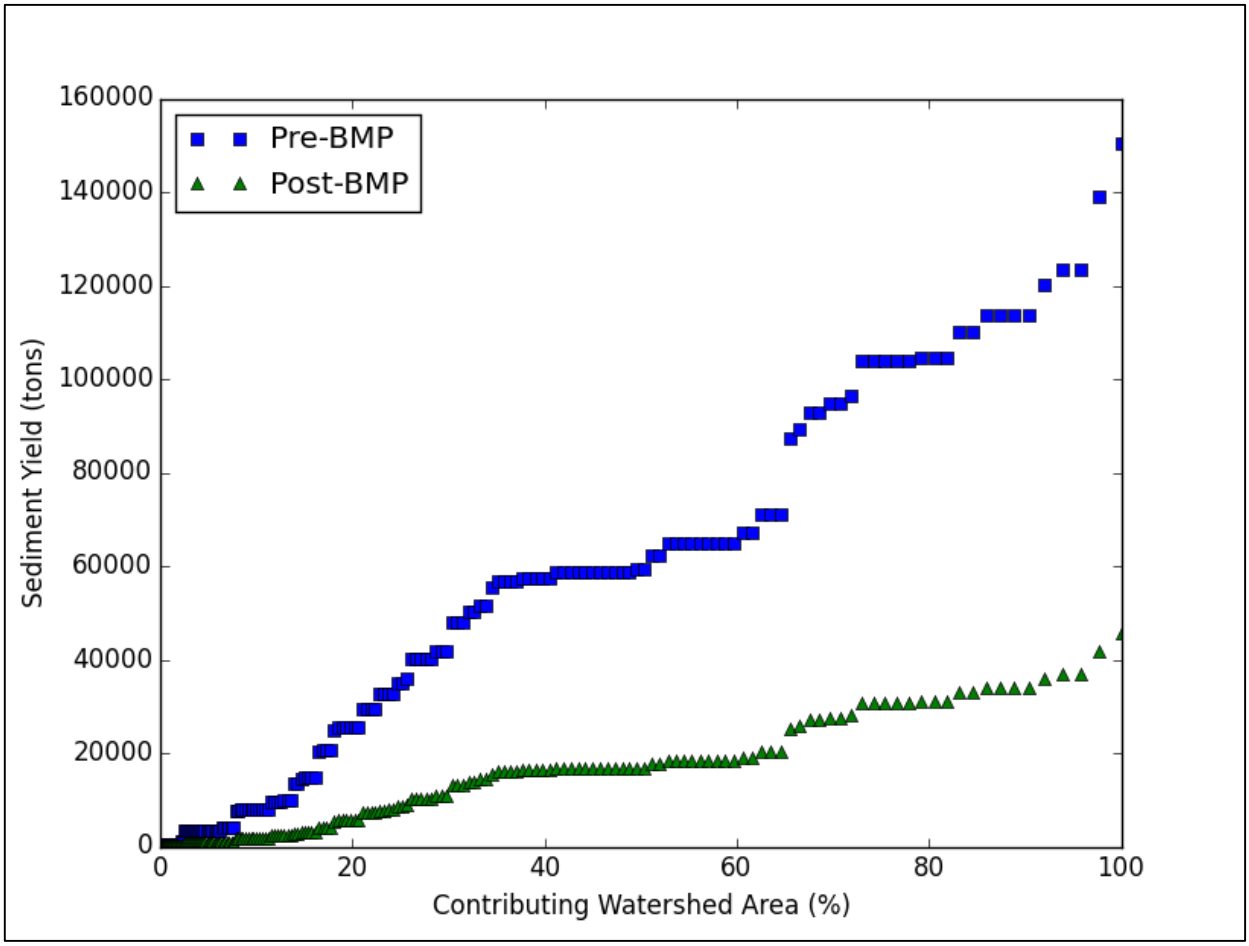

Figure 4.2. The cumulative graph for sediment variation over contributing area before and after simulation of CPs in L'Anguille River watershed.

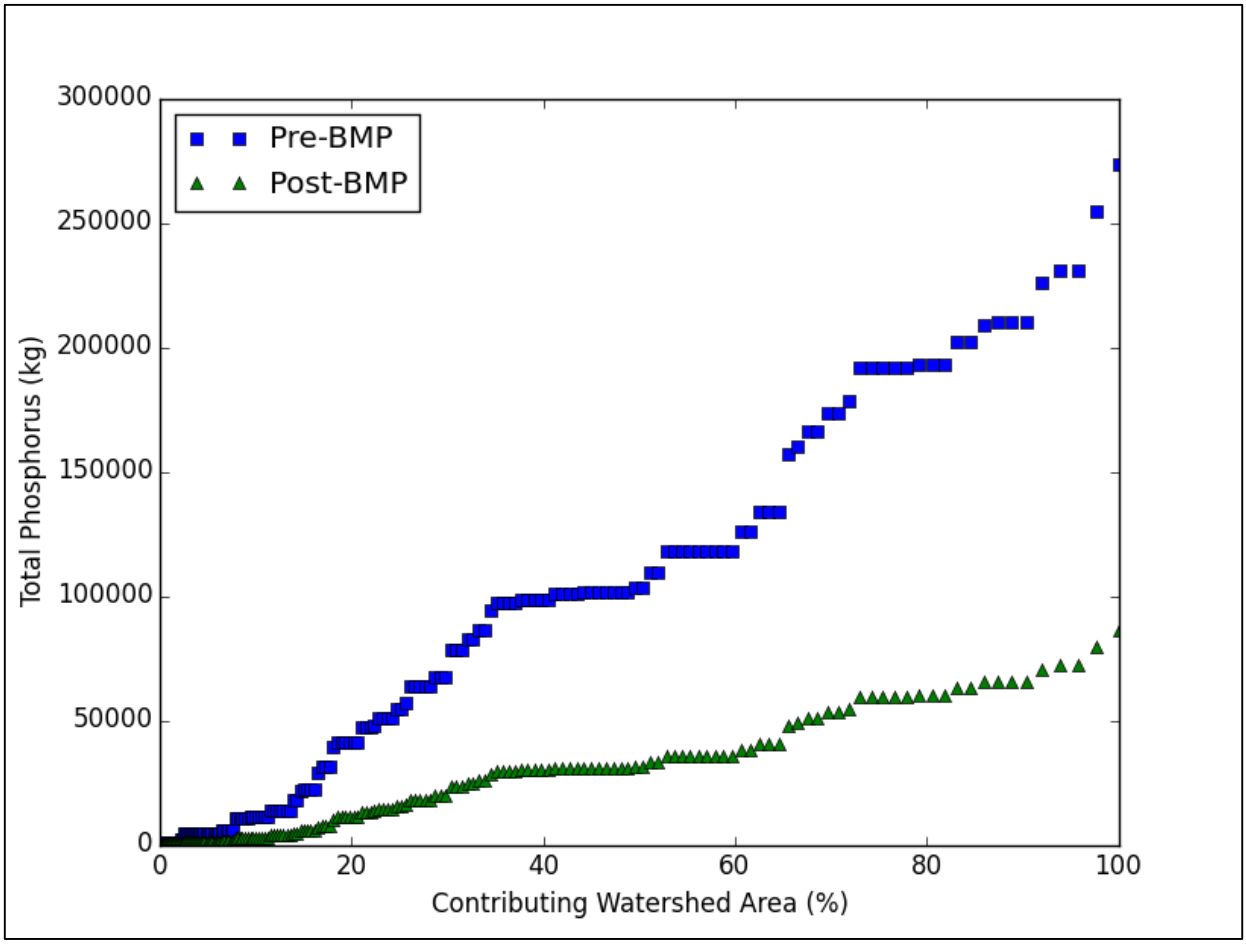

Figure 4.3. The cumulative graph for total phosphorus variation over contributing area before and after simulation of CPs in L'Anguille River watershed.

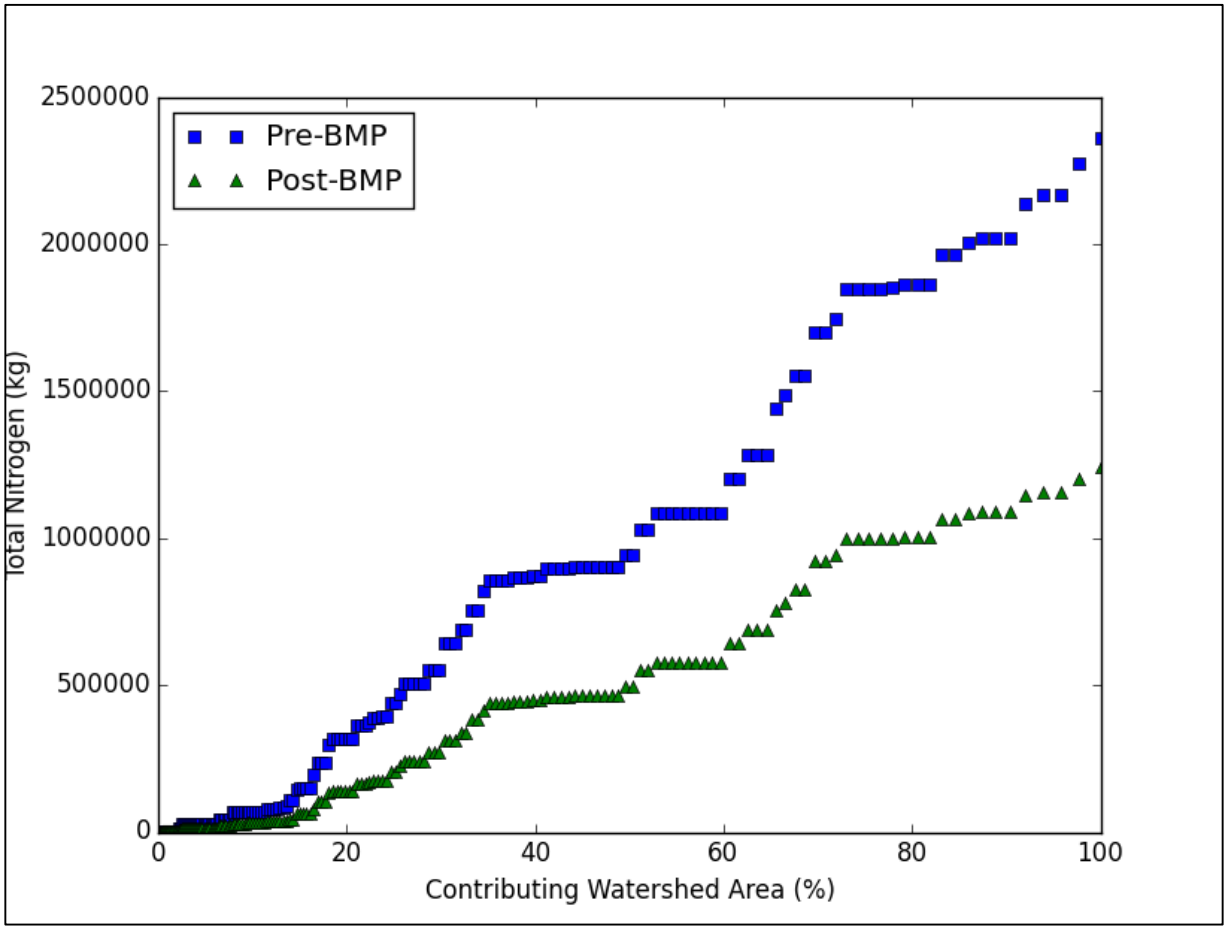

Figure 4.4. The cumulative graph for total nitrogen variation over contributing area before and after simulation of CPs in L'Anguille River watershed.

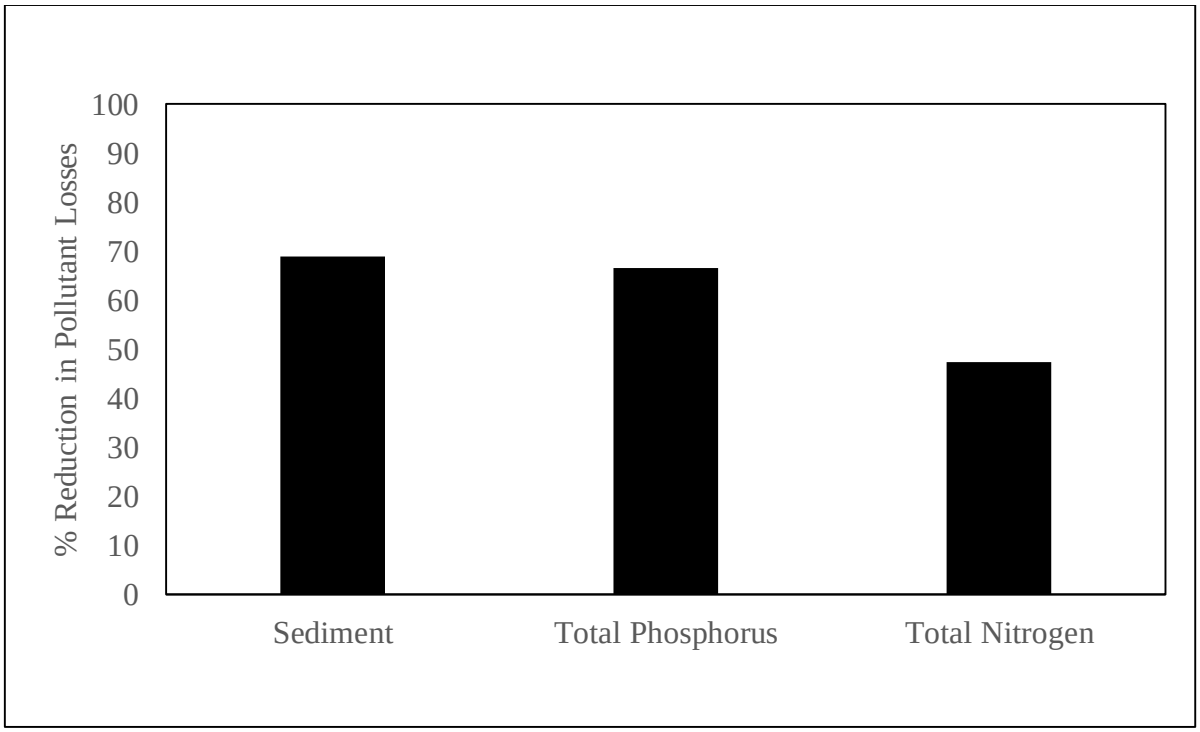

Figure 4.5. Percentage reduction in sediment, TP, and TN losses due to the simulation of filter strip and irrigation land leveling.

Table 4.2: Area-weighted average sediment, total phosphorus, and total nitrogen losses for the pre and post-CP scenario.

Output Variable	Pre-CP	Post-CP
Area-weighted average sediment yield (tons)	0.29	0.09
Area-weighted average total phosphorus (kg)	0.18	0.06
Area-weighted average total nitrogen (kg)	1.16	0.61

4.6 SUMMARY

Models, when combined with specialized software tools, can help decision makers to make decisions for CP implementations and controlling NPS pollution without the time requird to delve into the complexity of the models. An interactive Python-based desktop tool has been developed to simulate CPs on user-defined target areas in an already-developed SWAT model. The tool helps in interactive selection of target area as per the user needs. The tool was tested by simulating filter strip and irrigation land leveling in LRW. Simulating these CPs resulted in a reduction of 70% for sediment, 68% for TP, and 47% for TN losses. The test results illustrate how the tool can be applied to provide information regarding CP performance at selected sites at the targeted HRU level.

4.7 REFERENCES

ANRC. (2012). State of Arkansas Nutrient Reduction Strategy. Little Rock, Ark.: ANRC. Available at: https://static.ark.org/eeuploads/anrc/AR_Nutrient_Reduction_Strategy_101014.pdf. Accessed 26 October 2015.

Center for Advanced Spatial Technologies - CAST. (2007). Land use land cover: Fall 2006 (raster). Center for Advanced Spatial Technologies, Fayetteville, AR. Available at: http://www.geostor.arkansas.gov. Accessed 26 October 2015.

Demissie, Y., Yan, E., & Wu, M. (2012). Assessing regional hydrology and water quality implications of large-scale biofuel feedstock production in the Upper Mississippi river basin. *Environ. Science & Tech., 46*(16), 9174-9182.

Diebel, M. W., Maxted, J. T., Nowak, P. J., & Vander Zanden, M. J. (2008). Landscape planning for agricultural nonpoint source pollution reduction I: a geographical allocation framework. *Environ. Mgt., 42*(5), 789-802.

Kannan, N., Jeong, J., and Srinivasan, R. (2011). Hydrologic modeling of a canal-irrigated agricultural watershed with irrigation best management practices: Case study. *J. Hydrologic Eng., 16*(9), 746-757.

Niraula, R., Kalin, L., Wang, R., & Srivastava, P. (2012). Determining nutrient and sediment critical source areas with SWAT: effect of lumped calibration. *Trans. ASABE, 55*(1), 137-147.

Pai, N., Saraswat, D., & Daniels, M. (2011). Identifying priority subwatersheds in the Illinois River Drainage Area in Arkansas watershed using a distributed modeling approach. *Trans. ASABE, 54*(6), 2181-2196.

Pai, N., Saraswat, D., & Srinivasan, R. (2012). Field_SWAT: A tool for mapping SWAT output to field boundaries. *Computers and Geosci., 40*, 175-184.

Panagopoulos, Y., Makropoulos, C., Baltas, E., & Mimikou, M. (2011). SWAT parameterization for the identification of critical diffuse pollution source areas under data limitations. *Ecological Modelling, 222*(19), 3500-3512.

Parmenter, B. (2007). Creating a smaller data set from a larger dataset – vector data. Tufts university GIS tip sheet. Medford, MA.: Tufts University, Department of Urban and Environmental Policy and Planning.

Sarkar, S., Miller, S. A., Frederick, J. R., & Chamberlain, J. F. (2011). Modeling nitrogen loss from switchgrass agricultural systems. *Biomass and bioenergy*, *35*(10), 4381-4389.

Seaber, P., Kapinos, F. P., & Knapp, G. L. (1994). Hydrological Unit Maps. USGS water-supply Paper # 2294. Denver, CO: United States Geological Survey.

White, M., & Arnold, J. (2009). Development of a simplistic vegetative filter strip module for sediment and nutrient retention at the field scale. *Hydrological Processes, 23*, 1602-1616.

White, M. J., Storm, D. E., Busteed, P. R., Stoodley, S. H., & Phillips, S. H. (2009). Evaluating nonpoint source critical source area contributions at the watershed scale. *J. Environ. Qual., 38*(4), 1654-1663.

5. Summary And Recommendations

5.1 Synthesis of Chapter Contents

The overall goal of this research was to provide tools to assess impacts of CPs on target areas. Assessment of MRBI-recommended CPs (Chapter 2), CP tool for targeting CPs in a binary environment (Chapter 3) and in an interactive environment (Chapter 4) are some of the aspects of managing watersheds that were addressed using approaches based on the SWAT model. The tools developed in the dissertation are open source, free, and user-friendly. The following sections outline all the objectives and key results from each chapter.

5.1.1 Chapter 2

The objectives of Chapter 2 were to illustrate the process of assessing the predicted quantitative water quality impacts of selected MRBI-recommended CPs using SWAT in the LRW. Information from multiple LULC images were processed using remote sensing methods to incorporate missing land uses. In order to accurately quantify model output and retain all spatial data, no thresholds for land use, soil, or slope were used to create HRUs. For illustration pruposes, seven MRBI-recommended CPs were simulated in the SWAT model.

The results from this chapter can inform watershed planners and policy-makers to select and target appropriate CPs, which will most effectively bring about desired nutrient and sediment load reductions. Out of the MRBI CPs simulated in the LRW, critical area planting was found to be the most effective in reducing predicted nutrient losses (58% for TP and 16% for TN) and sediment losses (80%), followed by filter strip, irrigation land leveling, grade stabilization structure, irrigation pipeline, irrigation water management, and nutrient management.

It should be noted that the CPs should be selected and defined in the SWAT model with caution. A CP that is not suitable for the field or which cannot be well represented with SWAT parameters might report problematic and unreliable results relating to the impacts of CPs on water quality. The analyst should avoid CP modeling schemes that are purely empirical and applied outside of the context where they were developed.

5.1.2 CHAPTER 3

The objective of Chapter 3 was to develop a tool to help assess the effectiveness of CPs on user-defined target area with a graphical user interface developed for SWAT for targeting CPs. The tool was designed to simulate CPs at the lowest simulation level (i.e. HRU) of the SWAT model by building a new targeting procedure for SWAT applications and decision-making. Irrigation land leveling CP was pre-defined in the tool to illustrate the process.

The tool simulated the irrigation land leveling on the user defined target area. A decreasing trend for sediment, TP, and TN losses was observed with the simulation of irrigation land leveling. Simulating the CP resulted in a reduction of 22% for sediment, 20% for TP, and 12% for TN losses.

It should be noted that the target area should be thoroughly investigated by the user. A user should not simulate a CP where it is not required or where it has already been implemented. Moreover, the same CP (or the same applied model parameters) should not be implemented everywhere in the watershed. As some areas have different environmental and soil conditions, implementation of a uniform set of inputs across the whole area might not be a suitable approach for targeting CPs.

5.1.3 CHAPTER 4

The objective of Chapter 4 was to define the target area interactively in the CP tool to simulate CPs on the selected target area. A Python-based tool was developed in this study that uses past-developed SWAT models and simulates CPs at the user-defined locations using an interactive CP simulation approach. The tool was tested for LRW located in northeastern Arkansas. The filter strip and irrigation land leveling was simulated on the interactively-selected target area.

For the test, the tool allowed the user to select a target area interactively and predicted a 70% decrease in sediment losses, 68% decrease in TP losses, and 47% decrease in TN losses.

As the tool does not have the capability to sense the right CP for the right target area, the user should be aware of the target area conditions before defining the area interactively. The user should define both the target area and the CP.

5.2 FUTURE DIRECTIONS

1. In chapter 2, seven MRBI CPs were assessed for water quality impacts. In future, more CPs can be assessed for water quality. The parameters representing the CPs in the model should be tested rigorously. Some parameters in SWAT might not correctly represent the CPs to simulate the practices realistically.

2. In chapter 3 and chapter 4, a Python-based desktop version of the CP tool was developed to accurately analyze the impact of CPs on target area. In future, the desktop version could be converted into a web version. This would enable the user to upload the input data on the cloud and download the results without the need to install any software or related libraries.

3. The ability to simulate a set of fields with varying soil and crop conditions could be accomplished by further tool development that would allow input of multiple areas and specification by the user of unique input data and management practices for each.

APPENDIX A1. PACKAGES REQUIRED FOR INSTALLING THE CP TOOL

Native Python modules

tkinter - https://docs.python.org/3.4/library/tkinter.html?highlight=tkinter

os - https://docs.python.org/3.4/library/os.html?highlight=os#module-os

sys - https://docs.python.org/3.4/library/sys.html?highlight=sys#module-sys

shutil - https://docs.python.org/3.4/library/shutil.html?highlight=shutil#module-shutil

glob - https://docs.python.org/3.4/library/glob.html?highlight=glob#module-glob

fileinput - https://docs.python.org/3.4/library/fileinput.html?highlight=fileinput#module-fileinput

subprocess - https://docs.python.org/3.4/library/subprocess.html?highlight=subprocess#module-subprocess

time - https://docs.python.org/3.4/library/time.html?highlight=time#module-time

re - https://docs.python.org/3.4/library/re.html?highlight=re#module-re

Third party modules

pyqt4 - https://pypi.python.org/pypi/PyQt4

matplotlib - https://pypi.python.org/pypi/matplotlib/1.5.0

pillow (PIL) - https://pypi.python.org/pypi/Pillow/3.0.0

pylab - https://pypi.python.org/pypi/pylab/0.1.3

scipy - https://pypi.python.org/pypi/scipy/0.16.1

numpy - https://pypi.python.org/pypi/numpy/1.10.1

pyshp (shapefile) - https://pypi.python.org/pypi/pyshp/1.2.3

Other libraries

gdal_cookbook - https://pcjericks.github.io/py-gdalogr-cookbook/

gdal_translate - http://www.gdal.org/gdal_translate.html

gdalwarp - http://www.gdal.org/gdalwarp.html

APPENDIX A2. PYTHON CODES FOR THE BINARY TARGETED AREA PROCESS IN THE CP TOOL

```
#Packages
from PyQt4 import QtCore, QtGui, uic
from osgeo import gdal
from scipy.spatial import cKDTree
from tkinter import filedialog
from tkinter import messagebox
from PIL import Image
from pylab import *
import sys
sys.path.append(r'C:\Python34\Lib\site-packages\PIL')
import tkinter
import scipy
import numpy.ma as ma
import matplotlib.pyplot as plt
import shapefile
import os
import shutil
import numpy as np
import glob
import fileinput
import subprocess
import time
import re
import gdaltransformer
import gdal_cookbook

root = tkinter.Tk()
root.withdraw()

class MainWindow(QtGui.QMainWindow):

  #Initialising the GUI
   def __init__(self):
      QtGui.QMainWindow.__init__(self)
      self.ui = uic.loadUi(os.getcwd() + '\\cptool16.ui')
      self.ui.show()

      #Connecting the GUI with functions
      self.ui.actionSWAT_Project_Folder.triggered.connect(self.Select_SWAT_Project_Folder)
      self.ui.actionTargeted_Area.triggered.connect(self.Select_Targeted_Area)
      self.ui.actionProcess.triggered.connect(self.Select_Process)
      self.ui.actionRun_SWAT_Model.triggered.connect(self.Select_Run_SWAT_Model)
      self.ui.actionRun_SWAT_Model_2.triggered.connect(self.Select_Run_SWAT_Model_2)
      self.ui.actionCP1.triggered.connect(self.Select_actionCP1)
```

```
        self.ui.actionInput_Your_CP_Practice.triggered.connect(self.Select_Inputyourcp)
        self.ui.actionSed.triggered.connect(self.Create_Sed)
        self.ui.actionSolP.triggered.connect(self.Create_SolP)
        self.ui.actionNitN_Ground.triggered.connect(self.Create_NitN_Ground)
        self.ui.actionGraph2_3.triggered.connect(self.Create_Graph2)
        self.ui.actionManual.triggered.connect(self.help)

        #Defining global variables
        self.area_and_sed = []
        self.area_and_sed2 = []
        self.area_and_OrgP = []
        self.area_and_OrgP2 = []
        self.area_and_SedP = []
        self.area_and_SedP2 = []
        self.area_and_SolP = []
        self.area_and_SolP2 = []
        self.area_and_NitN = []
        self.area_and_NitN2 = []
        self.area_and_OrgN = []
        self.area_and_OrgN2 = []
        self.area_and_NitN_Lat = []
        self.area_and_NitN_Lat2 = []
        self.area_and_NitN_Ground = []
        self.area_and_NitN_Ground = []
        self.tp = []
        self.tp2 = []
        self.tn = []
        self.tn2 = []
        self.reduction1 = []
        self.reduction2 = []
        self.reduction3 = []
        self.sediment = []
        self.sediment2 = []
        self.phosphorus = []
        self.phosphorus2 = []
        self.nitrogen = []
        self.nitrogen2 = []
        self.click_counts = 0
        self.click_coordinates = []
        self.hrus1_filename = ""
        self.hrus_pixmap = ""
        self.iv_width = self.ui.ImageViewer.width()
        self.iv_height = self.ui.ImageViewer.height()
        self.ui.statusMessageLabel.setText("Please select your SWAT Project folder under the
Input menu")
        app.processEvents()
```

```
        #Adding logos to the GUI
        self.ui.logo.setPixmap(QtGui.QPixmap(
            os.getcwd() + "/images/logo.png").scaled(
                351, 81, QtCore.Qt.KeepAspectRatio))

        self.ui.logo2.setPixmap(QtGui.QPixmap(
            os.getcwd() + "/images/logo2.png").scaled(
                361, 81, QtCore.Qt.KeepAspectRatio))

    # defining SWAT project folder
    def Select_SWAT_Project_Folder(self):
        """asks the user to open SWAT project directory"""
        self.Project_Folder_Directory = filedialog.askdirectory(parent = root, title = "Select the
SWAT Project Folder")
        self.SWATDIR = self.Project_Folder_Directory
        self.ui.statusMessageLabel.setText("SWAT Project folder selected. Please select your
Target Area raster under the Input menu")
        self.hrus1_filename = self.SWATDIR + '/Watershed/Grid/hrus1'
        outfile = self.SWATDIR + '/Watershed/Grid/hrus1.jpg'
        subprocess.call(["gdal_translate", "-of", "JPEG", self.hrus1_filename, outfile])

        # Get outfile's dimensions and display outfile at full size
        im = Image.open(outfile)
        self.hrus_pixmap = QtGui.QPixmap(outfile).scaled(self.iv_width, self.iv_height,
QtCore.Qt.KeepAspectRatio)
        self.ui.ImageViewer.setPixmap(self.hrus_pixmap)
        self.ui.scrollArea.setWidget(self.ui.ImageViewer)
        app.processEvents()

    # defining targeted area
    def Select_Targeted_Area(self):
        self.Targeted_Area_Directory = filedialog.askopenfile(parent = root, title = "Select
Targeted Area")
        self.filename = self.Targeted_Area_Directory.name
        outfile = self.filename[:-3] + "jpg"
        subprocess.call(["gdal_translate", "-of", "JPEG", self.filename, outfile])

        # Get outfile's dimensions and display outfile at full size
        im = Image.open(outfile)
        self.ui.ImageViewer.setFixedHeight(im.size[1])
        self.ui.ImageViewer.setFixedWidth(im.size[0])
        self.ui.ImageViewer.setPixmap(QtGui.QPixmap(
            outfile).scaled(
                im.size[0], im.size[1], QtCore.Qt.KeepAspectRatio))
```

```
        self.ui.statusMessageLabel.setText("Targeted area selected. Please click Process under the
Processing menu")
        app.processEvents()

    def read_target_raster(self):
        '''Open the target raster with gdal and collect the raster's properties'''
        target_raster = gdal.Open(self.hrus1_filename)
        target_raster_gt = target_raster.GetGeoTransform()
        # Stores geographic coordinate tuples
        map_coordinates = []
        # Loop through each set of pixel coordinates (from user's clicks)
        for xy in self.click_coordinates:
            # Use pixelToMap to get the geographic coordinates and add tuple to list
            tx, ty = gdaltransformer.pixelToMap(xy[0], xy[1], target_raster_gt)
            map_coordinates.append((tx, ty))
        # Prints the results
        for xy in map_coordinates:
            # print(xy)
            self.create_mask_shapefile(map_coordinates)

    #Creating a mask shapefile
    def create_mask_shapefile(self, map_coordinates):
        w = shapefile.Writer(shapefile.POLYGON)
        map_parts = []
        for xy in map_coordinates:
            map_parts.append([xy[0], xy[1]])
        w.poly(parts=[map_parts])
        w.field('mask_name', 'C', '40')
        w.record('test')
        w.save(self.SWATDIR + '/Watershed/Grid/' + 'hrus1_mask.shp')
        self.create_masked_raster()

    #Creating a masked raster
    def create_masked_raster(self):
        hrus1_mask = self.SWATDIR + '/Watershed/Grid/hrus1_mask.shp'
        hrus1 = self.SWATDIR + '/Watershed/Grid/hrus1'
        outfile = self.SWATDIR + '/Watershed/Grid/hrus1_masked.tif'
        subprocess.call(["gdalwarp", "-of", "GTiff", "-cutline", hrus1_mask, "-cl", "hrus1_mask",
                         "-crop_to_cutline", hrus1, outfile])
        self.create_new_target_raster(hrus1, outfile)

    def create_new_target_raster(self, raster, raster_mask):
        # Create new raster from original raster where all cells
        # are equal to 0 or 255 (NoData)
        # First convert raster to array
        original_raster_array = gdal_cookbook.raster2array(raster)
```

```
        # Get no data value of array
        original_raster_no_data_value = gdal_cookbook.getNoDataValue(raster)

        # Update original raster - everything except NoData cells converted to "0"
        original_raster_array[original_raster_array != original_raster_no_data_value] = 0

        # Create new raster with updated array
        raster_new = self.SWATDIR + '/Watershed/Grid/hrus1_base.tif'
        gdal_cookbook.array2raster(raster, raster_new, original_raster_array)

        # Repeat same process for raster mask, except convert mask raster's cells to "1" - the target
areas
        raster_mask_array = gdal_cookbook.raster2array(raster_mask)
        raster_mask_array[raster_mask_array != original_raster_no_data_value] = 1
        raster_mask_new = self.SWATDIR + '/Watershed/Grid/hrus1_target_area.tif'
        target_tif = self.SWATDIR + '/Watershed/Grid/target.tif'
        gdal_cookbook.array2raster(raster_mask, raster_mask_new, raster_mask_array)
        #print(os.getcwd())

        # Now merge two rasters together to create target hru
        subprocess.Popen(["python", "gdal_merge.py", "-init", "255", "-o", target_tif, raster_new,
raster_mask_new])

    #defining HRU raster file
    def readHruRaster(self):
        rst_read = gdal.Open(self.SWATDIR + '/Watershed/Grid/hrus1')
        return rst_read.ReadAsArray()

    #defining HRU shape file
    def readHruShape(self):
        shp_read = shapefile.Reader(self.SWATDIR + '/Watershed/Shapes/hru1.shp')
        return np.asarray(shp_read.records(),dtype='a9')[:,0]

    #defining processing step
    def Select_Process(self):
        self.ui.statusMessageLabel.setText("Processing in progress")
        app.processEvents()

        '"process for post-threshold layer creation"'
        Post_Threshold = self.SWATDIR + '/Post_Threshold'
        HRU_shp = self.SWATDIR + '/Watershed/Shapes/hru1.shp'
        Pre_Thresh_HRU = self.SWATDIR + '/Watershed/Grid/hrus1'
        ThreshHRU = Post_Threshold + '/Raster/thresh_HRU'

        if os.path.exists(Post_Threshold):
```

```
            shutil.rmtree(Post_Threshold)
            os.makedirs(Post_Threshold)
        else:
            print("Post_Threshold folder does not exist within SWAT model folder")

        if os.path.exists(Post_Threshold + '/Raster'):
            shutil.rmtree(Post_Threshold + '/Raster')
            os.makedirs(Post_Threshold + '/Raster')
        else:
            os.makedirs(Post_Threshold + '/Raster')
        print("Post_Threshold folder structure created within your SWAT model folder...")

        thresholdApplied = 0

        rst_array = self.readHruRaster()
        hru_list = self.readHruShape()

        #Creates a new array with the same dimensions as the HRU raster array and initializes all
value to -99
        new_hru_array = np.zeros((len(rst_array), len(rst_array[0])), dtype='a9')
        zeroIdx = np.where(new_hru_array == b'')
        new_hru_array[zeroIdx] = b'-99'

        for hruIndex, hru in enumerate(hru_list):
            idx = np.where(rst_array == int(hru))
            new_hru_array[idx] = str(hruIndex + 1).encode()

        #Creating masks
        print("creating masks")

        fill_value = b'-99'
        new_hru_array = ma.masked_array(new_hru_array, new_hru_array == fill_value)

        # assigning values to the actually merged HRUs
        x, y = np.mgrid[0:new_hru_array.shape[0], 0:new_hru_array.shape[1]]
        xygood = np.array((x[~new_hru_array.mask], y[~new_hru_array.mask])).T
        xybad = np.array((x[new_hru_array.mask], y[new_hru_array.mask])).T
        #print("ckdetree")
        new_hru_array[new_hru_array.mask] =
new_hru_array[~new_hru_array.mask][cKDTree(xygood).query(xybad)[1]]
        print("ckdetree complete")
        ds1 = gdal.Open(self.filename)

        if ds1 is None:
            print('Could not open targeted file')
            sys.exit(1)
```

```
        targeted_array = np.array(ds1.GetRasterBand(1).ReadAsArray())
        tar_hru_array = np.array([[""]*len(new_hru_array[0])]*len(new_hru_array), dtype='a8')
        tIdx = np.where(targeted_array == 1)

        for i in range(0, len(new_hru_array)):
            for j in range(0, len(new_hru_array[0])):
                if targeted_array[i][j] == 1:
                    tar_hru_array[i][j] = new_hru_array[i][j] + b"t"
                else:
                    tar_hru_array[i][j] = new_hru_array[i][j]

            print("loop 1 ends")
            hru_range = list(set(tar_hru_array.flatten()))

        print("loop 2 starts")
        self.hru_dict = {}
        for hru in hru_range:
            t_count = len(np.where(tar_hru_array == hru + b"t")[0])
            hru_count = len(np.where(tar_hru_array == hru)[0])
            t_hru_ratio = float(t_count) / float(t_count + hru_count)
            if np.where(tar_hru_array == hru + b"t"):
                self.hru_dict[hru.split(b"t")[0]] = float(t_hru_ratio)
        print("loop 2 ends")
        print("Processing is complete.")

        self.ui.statusMessageLabel.setText("CP simulation in progress")
        app.processEvents()
        i = 0
        os.chdir(self.SWATDIR + '/Scenarios/Default/TxtInOut')
        for files in glob.glob("*.mgt"):
            for line in fileinput.input([files], inplace=True):
                line = line.replace("0.000    | FILTERW", "0.000    | FILTERW")
                sys.stdout.write(line)

        self.ui.statusMessageLabel.setText("Processing complete. Please click on Run SWAT
Model under the processing menu for pre-CP simulation")
        app.processEvents()

    #defining SWAT run for the pre-CP scenario
    def Select_Run_SWAT_Model(self):
        self.ui.statusMessageLabel.setText("SWAT model run in progress")
        app.processEvents()
        exe_str = self.SWATDIR + '/Scenarios/Default/TxtInOut/swat2012.exe'
        parent = subprocess.Popen(exe_str, stderr=subprocess.PIPE)
        time.sleep(5)
```

```
        self.ui.statusMessageLabel.setText("SWAT run complete. Please select your CP from the
Input CP Practice under the Scenario menu")
        app.processEvents()

        SWAT_output = self.SWATDIR + '/Scenarios/Default/TxtInOut'
        SWAT = open(SWAT_output+"\\output.hru", "r")
        SWATlines = SWAT.readlines()
        results = []

        #store the output for sediments
        months = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12']
        crops = ["PAST", "RNGE"]
        area_and_sed = []
        for a in range(9, len(SWATlines)):
            content = SWATlines[a]
            class MagicString(str):
                magicSplit = str.split
            s = MagicString(content)
            x = s.magicSplit()
            match = re.match(r"([a-z]+)([0-9]+)", x[0], re.I)
            if match:
                x = match.groups()
            area = x[5]
            # print(area.split('.'))
            if len(area.split('.')) == 3 and area.split('.')[0] in months and x[0] in crops:
                area = area.split('.')[2]
                # print("HERE Area")
                # print (area)
                tmp = (float("." + area), float(x[34])*self.hru_dict[bytes(x[1], 'utf-8')])
                area_and_sed.append(tmp)

        self.area_and_sed = sorted(area_and_sed, key=lambda area_and_sed: area_and_sed[0],
reverse=False)
        self.sediment = self.loss(self.area_and_sed)

        #store the output for organic phosphorus
        months = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12']
        area_and_OrgP = []
        for a in range(9, len(SWATlines)):
            content = SWATlines[a]
            class MagicString(str):
                magicSplit = str.split
            s = MagicString(content)
            x = s.magicSplit()
            match = re.match(r"([a-z]+)([0-9]+)", x[0], re.I)
            if match:
```

```
        x = match.groups()
    area = x[5]
    # print(area.split('.'))
    if len(area.split('.')) == 3 and area.split('.')[0] in months and x[0] in crops:
        area = area.split('.')[2]
        # print("HERE")
        tmp = (float("." + area), float(x[57])*self.hru_dict[bytes(x[1], 'utf-8')])
        area_and_OrgP.append(tmp)

#store the output for sediment phosphorus
months = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12']
area_and_SedP = []
for a in range(9, len(SWATlines)):
    content = SWATlines[a]
    class MagicString(str):
        magicSplit = str.split
    s = MagicString(content)
    x = s.magicSplit()
    match = re.match(r"([a-z]+)([0-9]+)", x[0], re.I)
    if match:
        x = match.groups()
    area = x[5]
    # print(area.split('.'))
    if len(area.split('.')) == 3 and area.split('.')[0] in months and x[0] in crops:
        area = area.split('.')[2]
        # print("HERE")
        tmp = (float("." + area), float(x[58])*self.hru_dict[bytes(x[1], 'utf-8')])
        area_and_SedP.append(tmp)

#store the output for soluble phosphorus
months = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12']
area_and_SolP = []
for a in range(9, len(SWATlines)):
    content = SWATlines[a]
    class MagicString(str):
        magicSplit = str.split
    s = MagicString(content)
    x = s.magicSplit()
    match = re.match(r"([a-z]+)([0-9]+)", x[0], re.I)
    if match:
        x = match.groups()
    area = x[5]
    # print(area.split('.'))
    if len(area.split('.')) == 3 and area.split('.')[0] in months and x[0] in crops:
        area = area.split('.')[2]
        # print("HERE")
```

```
        tmp = (float("." + area), float(x[63])*self.hru_dict[bytes(x[1], 'utf-8')])
        area_and_SolP.append(tmp)

#calculating total phosphorus
tp = list(np.array(area_and_OrgP) + np.array(area_and_SedP) + np.array(area_and_SolP))
self.tp = sorted(tp, key=lambda tp: tp[0], reverse=False)
self.phosphorus = self.loss(self.tp)

#store the output for nitrate-nitrogen
months = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12']
area_and_NitN = []
for a in range(9, len(SWATlines)):
    content = SWATlines[a]
    class MagicString(str):
        magicSplit = str.split
    s = MagicString(content)
    x = s.magicSplit()
    match = re.match(r"([a-z]+)([0-9]+)", x[0], re.I)
    if match:
        x = match.groups()
    area = x[5]
    # print(area.split('.'))
    if len(area.split('.')) == 3 and area.split('.')[0] in months and x[0] in crops:
        area = area.split('.')[2]
        # print("HERE")
        tmp = (float("." + area), float(x[59])*self.hru_dict[bytes(x[1], 'utf-8')])
        area_and_NitN.append(tmp)

#store the output for organic nitrogen
months = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12']
area_and_OrgN = []
for a in range(9, len(SWATlines)):
    content = SWATlines[a]
    class MagicString(str):
        magicSplit = str.split
    s = MagicString(content)
    x = s.magicSplit()
    match = re.match(r"([a-z]+)([0-9]+)", x[0], re.I)
    if match:
        x = match.groups()
    area = x[5]
    # print(area.split('.'))
    if len(area.split('.')) == 3 and area.split('.')[0] in months and x[0] in crops:
        area = area.split('.')[2]
        # print("HERE")
        tmp = (float("." + area), float(x[56])*self.hru_dict[bytes(x[1], 'utf-8')])
```

```
            area_and_OrgN.append(tmp)

        #store the output for lateral nitrate-nitrogen
        months = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12']
        area_and_NitN_Lat = []
        for a in range(9, len(SWATlines)):
            content = SWATlines[a]
            class MagicString(str):
                magicSplit = str.split
            s = MagicString(content)
            x = s.magicSplit()
            match = re.match(r"([a-z]+)([0-9]+)", x[0], re.I)
            if match:
                x = match.groups()
            area = x[5]
            # print(area.split('.'))
            if len(area.split('.')) == 3 and area.split('.')[0] in months and x[0] in crops:
                area = area.split('.')[2]
                # print("HERE")
                tmp = (float("." + area), float(x[61])*self.hru_dict[bytes(x[1], 'utf-8')])
                area_and_NitN_Lat.append(tmp)

        #store the output for groundwater nitrate-nitrogen
        months = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12']
        area_and_NitN_Ground = []
        for a in range(9, len(SWATlines)):
            content = SWATlines[a]
            class MagicString(str):
                magicSplit = str.split
            s = MagicString(content)
            x = s.magicSplit()
            match = re.match(r"([a-z]+)([0-9]+)", x[0], re.I)
            if match:
                x = match.groups()
            area = x[5]
            # print(area.split('.'))
            if len(area.split('.')) == 3 and area.split('.')[0] in months and x[0] in crops:
                area = area.split('.')[2]
                # print("HERE")
                tmp = (float("." + area), float(x[62])*self.hru_dict[bytes(x[1], 'utf-8')])
                area_and_NitN_Ground.append(tmp)

        #calculating the total nitrogen
        tn = list(np.array(area_and_NitN) + np.array(area_and_OrgN) +
np.array(area_and_NitN_Lat) + np.array(area_and_NitN_Ground))
        self.tn = sorted(tn, key=lambda tn: tn[0], reverse=False)
```

```
        self.nitrogen = self.loss(self.tn)

    #defining CPs
    def Select_Inputyourcp(self):
        messagebox.showwarning("Input your CP practice", "Open your SWAT project folder and
define CP practice of interest. \n Else, select predefined irrigation land leveling CP from the
tool.")

    def Select_actionCP1(self):
        self.ui.statusMessageLabel.setText("CP simulation in progress")
        app.processEvents()
        i = 0
        os.chdir(self.SWATDIR + '/Scenarios/Default/TxtInOut')

    #Defining irrigation land leveling CP
        for files in glob.glob("*.hru"):
            for line in fileinput.input([files], inplace=1):
                if "HRU_SLP" in line:
                    new_value = float(re.match(r'[-+]?[0-9]*\.?[0-9]+', line.lstrip()).group()) * .9
                    p = re.compile(r'[-+]?[0-9]*\.?[0-9]+')
                    filterw_line = p.sub(str(new_value), line)
                    sys.stdout.write(line.replace(line, filterw_line))
                else:
                    sys.stdout.write(line)
            fileinput.close()

        for files in glob.glob("*.hru"):
            for line in fileinput.input([files], inplace=1):
                if "SLSUBBSN" in line:
                    new_value = float(re.match(r'[-+]?[0-9]*\.?[0-9]+', line.lstrip()).group()) * .1
                    p = re.compile(r'[-+]?[0-9]*\.?[0-9]+')
                    filterw_line = p.sub(str(new_value), line)
                    sys.stdout.write(line.replace(line, filterw_line))
                else:
                    sys.stdout.write(line)
            fileinput.close()

        self.ui.statusMessageLabel.setText("Selected CP simulated. Please click on Run SWAT
Model under the Scenario menu for the post-CP simulation")
        app.processEvents()

    #defining SWAT run for the post-CP scenario
    def Select_Run_SWAT_Model_2(self):
        self.ui.statusMessageLabel.setText("SWAT model run for post-CP scenario in progress")
        app.processEvents()
        exe_str = self.SWATDIR + '/Scenarios/Default/TxtInOut/swat2012.exe'
```

```
        parent = subprocess.Popen(exe_str, stderr=subprocess.PIPE)
        time.sleep(5)
        self.ui.statusMessageLabel.setText("Please click on Sediment under the Graphs menu")
        app.processEvents()

        SWAT_output = self.SWATDIR + '/Scenarios/Default/TxtInOut'
        SWAT = open(SWAT_output+"\\output.hru", "r")
        SWATlines = SWAT.readlines()

        #store the output for post-CP sediment
        months = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12']
        crops = ["RNGE", "PAST", "FRSD"]
        area_and_sed2 = []

        for a in range(9, len(SWATlines)):
            content = SWATlines[a]
            class MagicString(str):
                magicSplit = str.split
            s2 = MagicString(content)
            x2 = s2.magicSplit()
            match = re.match(r"([a-z]+)([0-9]+)", x2[0], re.I)
            if match:
                x2 = match.groups()
            area2 = x2[5]

            if len(area2.split('.')) == 3 and area2.split('.')[0] in months and x2[0] in crops:
                area2 = area2.split('.')[2]

                tmp = (float("." + area2), float(x2[34])*self.hru_dict[bytes(x2[1], 'utf-8')])
                area_and_sed2.append(tmp)

        self.area_and_sed2 = sorted(area_and_sed2, key=lambda area_and_sed2: area_and_sed2[0],
reverse=False)
        self.sediment2 = self.loss(self.area_and_sed2)

        #store the output for post-CP organic phosphorus
        months = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12']
        area_and_OrgP2 = []
        for a in range(9, len(SWATlines)):
            content = SWATlines[a]
            class MagicString(str):
                magicSplit = str.split
            s2 = MagicString(content)
            x2 = s2.magicSplit()
            match = re.match(r"([a-z]+)([0-9]+)", x2[0], re.I)
            if match:
```

```
        x2 = match.groups()
    area2 = x2[5]

    if len(area2.split('.')) == 3 and area2.split('.')[0] in months and x2[0] in crops:
        area2 = area2.split('.')[2]

        tmp = (float("." + area2), float(x2[57])*self.hru_dict[bytes(x2[1], 'utf-8')])
        area_and_OrgP2.append(tmp)

#store the output for post-CP sediment phosphorus
months = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12']
area_and_SedP2 = []
for a in range(9, len(SWATlines)):
    content = SWATlines[a]
    class MagicString(str):
        magicSplit = str.split
    s2 = MagicString(content)
    x2 = s2.magicSplit()
    match = re.match(r"([a-z]+)([0-9]+)", x2[0], re.I)
    if match:
        x2 = match.groups()
    area2 = x2[5]

    if len(area2.split('.')) == 3 and area2.split('.')[0] in months and x2[0] in crops:
        area2 = area2.split('.')[2]

        tmp = (float("." + area2), float(x2[58])*self.hru_dict[bytes(x2[1], 'utf-8')])
        area_and_SedP2.append(tmp)

#store the output for post-CP soluble phosphorus
months = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12']
area_and_SolP2 = []

for a in range(9, len(SWATlines)):
    content = SWATlines[a]
    class MagicString(str):
        magicSplit = str.split
    s2 = MagicString(content)
    x2 = s2.magicSplit()
    match = re.match(r"([a-z]+)([0-9]+)", x2[0], re.I)
    if match:
        x2 = match.groups()
    area2 = x2[5]

    if len(area2.split('.')) == 3 and area2.split('.')[0] in months and x2[0] in crops:
        area2 = area2.split('.')[2]
```

```
            tmp = (float("." + area2), float(x2[63])*self.hru_dict[bytes(x2[1], 'utf-8')])
            area_and_SolP2.append(tmp)

        #Calculating total phosphorus for post-CP scenario
        tp2 = list(np.array(area_and_OrgP2) + np.array(area_and_SedP2) +
np.array(area_and_SolP2))
        self.tp2 = sorted(tp2, key=lambda tp2: tp2[0], reverse=False)
        self.phosphorus2 = self.loss(self.tp2)

        #store the output for post-CP nitrate-nitrogen
        months = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12']
        area_and_NitN2 = []
        for a in range(9, len(SWATlines)):
            content = SWATlines[a]
            class MagicString(str):
                magicSplit = str.split
            s2 = MagicString(content)
            x2 = s2.magicSplit()
            match = re.match(r"([a-z]+)([0-9]+)", x2[0], re.I)
            if match:
                x2 = match.groups()
            area2 = x2[5]

            if len(area2.split('.')) == 3 and area2.split('.')[0] in months and x2[0] in crops:
                area2 = area2.split('.')[2]

                tmp = (float("." + area2), float(x2[59])*self.hru_dict[bytes(x2[1], 'utf-8')])
                area_and_NitN2.append(tmp)

        #store the output for post-CP organic nitrogen
        months = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12']
        area_and_OrgN2 = []
        for a in range(9, len(SWATlines)):
            content = SWATlines[a]
            class MagicString(str):
                magicSplit = str.split
            s2 = MagicString(content)
            x2 = s2.magicSplit()
            match = re.match(r"([a-z]+)([0-9]+)", x2[0], re.I)
            if match:
                x2 = match.groups()
            area2 = x2[5]

            if len(area2.split('.')) == 3 and area2.split('.')[0] in months and x2[0] in crops:
                area2 = area2.split('.')[2]
```

```
            tmp = (float("." + area2), float(x2[56])*self.hru_dict[bytes(x2[1], 'utf-8')])
            area_and_OrgN2.append(tmp)

        #store the output for post-CP ateral nitrate-nitrogen
        months = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12']
        area_and_NitN_Lat2 = []
        for a in range(9, len(SWATlines)):
            content = SWATlines[a]
            class MagicString(str):
                magicSplit = str.split
            s2 = MagicString(content)
            x2 = s2.magicSplit()
            match = re.match(r"([a-z]+)([0-9]+)", x2[0], re.I)
            if match:
                x2 = match.groups()
            area2 = x2[5]

            if len(area2.split('.')) == 3 and area2.split('.')[0] in months and x2[0] in crops:
                area2 = area2.split('.')[2]

                tmp = (float("." + area2), float(x2[61])*self.hru_dict[bytes(x2[1], 'utf-8')])
                area_and_NitN_Lat2.append(tmp)

        #store the output for post-CP groundwater nitrate-nitrogen
        months = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12']
        area_and_NitN_Ground2 = []
        for a in range(9, len(SWATlines)):
            content = SWATlines[a]
            class MagicString(str):
                magicSplit = str.split
            s2 = MagicString(content)
            x2 = s2.magicSplit()
            match = re.match(r"([a-z]+)([0-9]+)", x2[0], re.I)
            if match:
                x2 = match.groups()
            area2 = x2[5]

            if len(area2.split('.')) == 3 and area2.split('.')[0] in months and x2[0] in crops:
                area2 = area2.split('.')[2]

                tmp = (float("." + area2), float(x2[62])*self.hru_dict[bytes(x2[1], 'utf-8')])
                area_and_NitN_Ground2.append(tmp)

        tn2 = list(np.array(area_and_NitN2) + np.array(area_and_OrgN2) +
np.array(area_and_NitN_Lat2) + np.array(area_and_NitN_Ground2))
```

```
    self.tn2 = sorted(tn2, key=lambda tn2: tn2[0], reverse=False)
    self.nitrogen2 = self.loss(self.tn2)

#defining graph
def Create_Sed(self):
    self.ui.statusMessageLabel.setText("Sediment Yield Graph Displayed")
    app.processEvents()
    x = []
    y = []
    z = []
    for i in range(0, len(self.area_and_sed)):
        x.append(self.area_and_sed[i][0])
        y.append(self.sediment[i])
        z.append(self.sediment2[i])
    xaccum = list(self.accumu(x))
    yaccum = list(self.accumu(y))
    zaccum = list(self.accumu(z))

    xaccu = list(self.accumu(x))
    yaccu = list(self.accu(y))
    zaccu = list(self.accu(z))

    xy = []

    sumxy = sum(y)
    sumx = sum(x)
    areawtave_pre = sumxy/(100*sumx)
    print ('Area-weighted ave. Sediment Yield (tons): Pre-CP = ' + str(areawtave_pre))

    xz = []

    sumxz = sum(z)
    sumx = sum(x)
    areawtave_post = sumxz/(100*sumx)
    print ('Area-weighted ave. Sediment Yield (tons): Post-CP = ' + str(areawtave_post))
    self.reduction1 = ((areawtave_post - areawtave_pre)/areawtave_pre)*100

    p1, = plt.plot(xaccu, yaccu, 'bs')
    p2, = plt.plot(xaccu, zaccu, 'g^')
    plt.legend([p1, p2], ["Pre-CP", "Post-CP"], loc=2)

    plt.ylabel('Sediment Yield (tons)')
    plt.xlabel('Contributing Watershed Area (%)')

    plt.show()
```

```
# defining graph
def Create_SolP(self):
    self.ui.statusMessageLabel.setText("TP Graph Displayed")
    app.processEvents()
    x = []
    y = []
    z = []
    for i in range(0, len(self.tp)):
        x.append(self.tp[i][0])
        y.append(self.phosphorus[i])
        z.append(self.phosphorus2[i])
    xaccum = list(self.accumu(x))
    yaccum = list(self.accumu(y))
    zaccum = list(self.accumu(z))

    xaccu = list(self.accumu(x))
    yaccu = list(self.accu(y))
    zaccu = list(self.accu(z))

    xy = []

    sumxy = sum(y)
    sumx = sum(x)
    areawtave_pre = sumxy/(100*sumx)
    print ('Area-weighted ave. Total Phosphorus (kg): Pre-CP = ' + str(areawtave_pre))

    xz = []

    sumxz = sum(z)
    sumx = sum(x)
    areawtave_post = sumxz/(100*sumx)
    print ('Area-weighted ave. Total Phosphorus (kg): Post-CP = ' + str(areawtave_post))

    self.reduction2 = ((areawtave_post - areawtave_pre)/areawtave_pre)*100

    p1, = plt.plot(xaccu, yaccu, 'bs')
    p2, = plt.plot(xaccu, zaccu, 'g^')
    plt.legend([p1, p2], ["Pre-CP", "Post-CP"], loc=2)

    plt.ylabel('Total Phosphorus (kg)')
    plt.xlabel('Contributing Watershed Area (%)')
    plt.show()

# defining graph
def Create_NitN_Ground(self):
    self.ui.statusMessageLabel.setText("TN Graph Displayed")
```

```
    app.processEvents()
    x = []
    y = []
    z = []
    for i in range(0, len(self.tn)):
        x.append(self.tn[i][0])
        y.append(self.nitrogen[i])
        z.append(self.nitrogen2[i])
    xaccum = list(self.accumu(x))
    yaccum = list(self.accumu(y))
    zaccum = list(self.accumu(z))

    xaccu = list(self.accumu(x))
    yaccu = list(self.accu(y))
    zaccu = list(self.accu(z))

    xy = []

    sumxy = sum(y)
    sumx = sum(x)
    areawtave_pre = sumxy/(100*sumx)
    print ('Area-weighted ave. Total Nitrogen (kg): Pre-CP = ' + str(areawtave_pre))

    xz = []

    sumxz = sum(z)
    sumx = sum(x)
    areawtave_post = sumxz/(100*sumx)
    print ('Area-weighted ave. Total Nitrogen (kg): Post-CP = ' + str(areawtave_post))

    self.reduction3 = ((areawtave_post - areawtave_pre)/areawtave_pre)*100

    p1, = plt.plot(xaccu, yaccu, 'bs')
    p2, = plt.plot(xaccu, zaccu, 'g^')
    plt.legend([p1, p2], ["Pre-CP", "Post-CP"], loc=2)

    plt.ylabel('Total Nitrogen (kg)')
    plt.xlabel('Contributing Watershed Area (%)')
    plt.show()

# defining graph
def Create_Graph2(self):
    N = 1
    ind = np.arange(N) # the x locations for the groups
    offset = 0.05
    width = 0.24    # the width of the bars
```

```
    fig = plt.figure()
    ax = fig.add_subplot(111)

    baseline = [self.reduction1]
    kwargs = {"hatch":'x'}
    rects1 = ax.bar(offset+ind, baseline, width, color='w', ecolor='k', **kwargs)

    leaf = [self.reduction2]
    kwargs = {"hatch":'.'}
    rects2 = ax.bar(offset+ind+width, leaf, width, color='w', ecolor='k', **kwargs)

    ultrapeer = [self.reduction3]
    kwargs = {"hatch":'/'}
    rects3 = ax.bar(offset+ind+width+width, ultrapeer, width, color='w', ecolor='k', **kwargs)

    # add labels
    ax.set_ylabel('% Reduction in Pollutant Losses')
    ax.set_xticks(offset+ind+width+width/3)

    ax.legend( (rects1[0], rects2[0], rects3[0]), ('Sediment', 'TP', 'TN'), loc = 'best')
    plt.tick_params(
        axis='x',          # changes apply to the x-axis
        which='both',      # both major and minor ticks are affected
        bottom='off',      # ticks along the bottom edge are off
        top='off',         # ticks along the top edge are off
        labelbottom='off') # labels along the bottom edge are off
    plt.show()

# defining cumulative function
def accu(self, list):
    total = 0
    for x in list:
        total += x
        yield total

# defining cumulative function as percentage
def accumu(self, list):
    total = 0
    for x in list:
        b = sum(list)
        total += x
        yield float(float(total)/float(b))*100

def loss(self, l):
    results = []
```

```
        for x in l:
            results.append(x[0]*x[1]*100)
        return results

    # defining help function
    def help(self):
        self.ui.statusMessageLabel.setText("Help Menu")
        app.processEvents()
        path_to_notepad = 'C:\\Program Files\\Microsoft Office\\Office15\\WINWORD.exe'
        path_to_file = 'C:\\temp\\CP Tool
Testing\\CP_Tool_Python34\\Code\\CPTool_Singh_Outline_28thJan2015.doc'
        subprocess.call([path_to_notepad, path_to_file])

if __name__ == "__main__":
    app = QtGui.QApplication(sys.argv)
    win = MainWindow()
    sys.exit(app.exec_())
```

APPENDIX A3. PYTHON CODES FOR THE INTERACTIVE TARGETED AREA PROCESS IN THE CP TOOL

```
#Packages
from PyQt4 import QtCore, QtGui, uic
from osgeo import gdal
from scipy.spatial import cKDTree
from tkinter import filedialog
from tkinter import messagebox
from PIL import Image
from pylab import *
import sys
sys.path.append(r'C:\Python34\Lib\site-packages\PIL')
import tkinter
import scipy
import numpy.ma as ma
import matplotlib.pyplot as plt
import shapefile
import os
import shutil
import numpy as np
import glob
import fileinput
import subprocess
import time
import re
import gdaltransformer
import gdal_cookbook

root = tkinter.Tk()
root.withdraw()

class MainWindow(QtGui.QMainWindow):

#defining global variables
    def __init__(self):
        QtGui.QMainWindow.__init__(self)
        self.ui = uic.loadUi(os.getcwd() + '\\cptool16.ui')
        self.ui.show()
        self.ui.actionSWAT_Project_Folder.triggered.connect(self.Select_SWAT_Project_Folder)
        self.ui.actionTargeted_Area.triggered.connect(self.Select_Targeted_Area)
        self.ui.actionProcess.triggered.connect(self.Select_Process)
        self.ui.actionRun_SWAT_Model.triggered.connect(self.Select_Run_SWAT_Model)
        self.ui.actionRun_SWAT_Model_2.triggered.connect(self.Select_Run_SWAT_Model_2)
        self.ui.actionCP1.triggered.connect(self.Select_actionCP1)
        self.ui.actionInput_Your_CP_Practice.triggered.connect(self.Select_Inputyourcp)
```

```
        self.ui.actionSed.triggered.connect(self.Create_Sed)
        self.ui.actionSolP.triggered.connect(self.Create_SolP)
        self.ui.actionNitN_Ground.triggered.connect(self.Create_NitN_Ground)
        self.ui.actionGraph2_3.triggered.connect(self.Create_Graph2)
        self.ui.actionManual.triggered.connect(self.help)
        self.scaleFactor = 1
        self.area_and_sed = []
        self.area_and_sed2 = []
        self.area_and_OrgP = []
        self.area_and_OrgP2 = []
        self.area_and_SedP = []
        self.area_and_SedP2 = []
        self.area_and_SolP = []
        self.area_and_SolP2 = []
        self.area_and_NitN = []
        self.area_and_NitN2 = []
        self.area_and_OrgN = []
        self.area_and_OrgN2 = []
        self.area_and_NitN_Lat = []
        self.area_and_NitN_Lat2 = []
        self.area_and_NitN_Ground = []
        self.area_and_NitN_Ground = []
        self.tp = []
        self.tp2 = []
        self.tn = []
        self.tn2 = []
        self.reduction1 = []
        self.reduction2 = []
        self.reduction3 = []
        self.sediment = []
        self.sediment2 = []
        self.phosphorus = []
        self.phosphorus2 = []
        self.nitrogen = []
        self.nitrogen2 = []
        self.click_counts = 0
        self.click_coordinates = []
        self.hrus1_filename = ""
        self.hrus_pixmap = ""
        self.iv_width = self.ui.ImageViewer.width()
        self.iv_height = self.ui.ImageViewer.height()
        self.ui.statusMessageLabel.setText("Please select your SWAT Project folder under the
Input menu")
        app.processEvents()

        # defining logo for the GUI
```

```
        self.ui.logo.setPixmap(QtGui.QPixmap(
           os.getcwd() + "/images/logo.png").scaled(
              351, 81, QtCore.Qt.KeepAspectRatio))

        self.ui.logo2.setPixmap(QtGui.QPixmap(
           os.getcwd() + "/images/logo2.png").scaled(
              361, 81, QtCore.Qt.KeepAspectRatio))

#defining SWAT project folder
    def Select_SWAT_Project_Folder(self):
        self.Project_Folder_Directory = filedialog.askdirectory(parent = root, title = "Select the
SWAT Project Folder")
        self.SWATDIR = self.Project_Folder_Directory
        self.ui.statusMessageLabel.setText("SWAT Project folder selected. Please select your
Target Area raster under the Input menu")
        self.hrus1_filename = self.SWATDIR + '/Watershed/Grid/hrus1'
        outfile = self.SWATDIR + '/Watershed/Grid/hrus1.jpg'

        subprocess.call(["gdal_translate", "-of", "JPEG", self.hrus1_filename, outfile])

        # Get outfile's dimensions and display outfile at full size
        im = Image.open(outfile)

        self.hrus_pixmap = QtGui.QPixmap(outfile).scaled(self.iv_width, self.iv_height,
QtCore.Qt.KeepAspectRatio)
        self.ui.ImageViewer.setPixmap(self.hrus_pixmap)
        self.ui.scrollArea.setWidget(self.ui.ImageViewer)
        # Call get_click_position method for mouse clicks on ImageViewer
        self.ui.ImageViewer.mousePressEvent = self.get_click_position
        app.processEvents()

#defining targeted area
    def Select_Targeted_Area(self):
        self.Targeted_Area_Directory = filedialog.askopenfile(parent = root, title = "Select
Targeted Area")
        self.filename = self.Targeted_Area_Directory.name
        outfile = self.filename[:-3] + "jpg"
        subprocess.call(["gdal_translate", "-of", "JPEG", self.filename, outfile])

        # Get outfile's dimensions and display outfile at full size
        im = Image.open(outfile)
        self.ui.ImageViewer.setFixedHeight(im.size[1])
        self.ui.ImageViewer.setFixedWidth(im.size[0])
        self.ui.ImageViewer.setPixmap(QtGui.QPixmap(
           outfile).scaled(
              im.size[0], im.size[1], QtCore.Qt.KeepAspectRatio))
```

```
        self.ui.statusMessageLabel.setText("Targeted area selected. Please click Process under the
Processing menu")
        app.processEvents()

    #def paintEvent(self, event):
        #print("paintEvent called")

    # Collect pixel coordinates from mouse clicks on ImageViewer
    def get_click_position(self, event):
        # If the user has not clicked at least three times...
        if self.click_counts != 3:
            # Store the current click's x,y and increase the click counter
            self.click_coordinates.append((event.pos().x(), event.pos().y()))
            self.click_counts += 1
        # User has clicked 4 times
        else:
            # Store the current (fourth) click's x,y and call the method
            # that converts pixel coordinates to geographic coordinates
            self.click_coordinates.append((event.pos().x(), event.pos().y()))

            points = [
                QtCore.QPoint(self.click_coordinates[0][0], self.click_coordinates[0][1]),
                QtCore.QPoint(self.click_coordinates[1][0], self.click_coordinates[1][1]),
                QtCore.QPoint(self.click_coordinates[2][0], self.click_coordinates[2][1]),
                QtCore.QPoint(self.click_coordinates[3][0], self.click_coordinates[3][1])
            ]

            painter = QtGui.QPainter(self.hrus_pixmap)
            painter.setBrush(QtGui.QBrush(QtCore.Qt.BDiagPattern))
            painter.drawPolygon(QtGui.QPolygon(points))

            #self.draw_rectangle()
            self.read_target_raster()

    def get_end_click_position(self, event):
        self.endx = event.x()
        self.endy = event.y()
        #print(self.endx)
        #print(self.endy)
        newLine = LineDraw(QtCore.QPoint(self.startx, self.starty), QtCore.Point(self.endx,
self.endy))

    # Convert user selected points from pixel coordinates to geographic coordinates
    def read_target_raster(self):
        # Open the target raster with gdal and collect the raster's properties
```

```
    target_raster = gdal.Open(self.hrus1_filename)
    target_raster_gt = target_raster.GetGeoTransform()
    # Stores geographic coordinate tuples
    map_coordinates = []
    # Loop through each set of pixel coordinates (from user's clicks)
    for xy in self.click_coordinates:
        # Use pixelToMap to get the geographic coordinates and add tuple to list
        tx, ty = gdaltransformer.pixelToMap(xy[0], xy[1], target_raster_gt)
        map_coordinates.append((tx, ty))
    # Prints the results
    for xy in map_coordinates:
        print(xy)
    self.create_mask_shapefile(map_coordinates)

#Creating a mask shapefile
def create_mask_shapefile(self, map_coordinates):
    w = shapefile.Writer(shapefile.POLYGON)
    map_parts = []
    for xy in map_coordinates:
        map_parts.append([xy[0], xy[1]])
    w.poly(parts=[map_parts])
    w.field('mask_name', 'C', '40')
    w.record('test')
    w.save(self.SWATDIR + '/Watershed/Grid/' + 'hrus1_mask.shp')
    self.create_masked_raster()

#Creating a masked raster
def create_masked_raster(self):
    hrus1_mask = self.SWATDIR + '/Watershed/Grid/hrus1_mask.shp'
    hrus1 = self.SWATDIR + '/Watershed/Grid/hrus1'
    outfile = self.SWATDIR + '/Watershed/Grid/hrus1_masked.tif'
    subprocess.call(["gdalwarp", "-of", "GTiff", "-cutline", hrus1_mask, "-cl", "hrus1_mask",
                     "-crop_to_cutline", hrus1, outfile])
    self.create_new_target_raster(hrus1, outfile)

def create_new_target_raster(self, raster, raster_mask):
    # Create new raster from original raster where all cells
    # are equal to 0 or 255 (NoData)

    # First convert raster to array
    original_raster_array = gdal_cookbook.raster2array(raster)
    # Get no data value of array
    original_raster_no_data_value = gdal_cookbook.getNoDataValue(raster)
    # Update original raster - everything except NoData cells converted to "0"
    original_raster_array[original_raster_array != original_raster_no_data_value] = 0
    # Create new raster with updated array
```

```
        raster_new = self.SWATDIR + '/Watershed/Grid/hrus1_base.tif'
        gdal_cookbook.array2raster(raster, raster_new, original_raster_array)
        # Repeat same process for raster mask, except convert mask raster's cells to "1" - the target
areas
        raster_mask_array = gdal_cookbook.raster2array(raster_mask)
        raster_mask_array[raster_mask_array != original_raster_no_data_value] = 1
        raster_mask_new = self.SWATDIR + '/Watershed/Grid/hrus1_target_area.tif'
        target_tif = self.SWATDIR + '/Watershed/Grid/target.tif'
        gdal_cookbook.array2raster(raster_mask, raster_mask_new, raster_mask_array)
        print(os.getcwd())
        # Now merge two rasters together to create target hru
        subprocess.Popen(["python", "gdal_merge.py", "-init", "255", "-o", target_tif, raster_new,
raster_mask_new])

    #defining HRU raster file
    def readHruRaster(self):
        rst_read = gdal.Open(self.SWATDIR + '/Watershed/Grid/hrus1')
        return rst_read.ReadAsArray()

    #defining HRU shape file
    def readHruShape(self):
        shp_read = shapefile.Reader(self.SWATDIR + '/Watershed/Shapes/hru1.shp')
        return np.asarray(shp_read.records(),dtype='a9')[:,0]

    #defining processing step
    def Select_Process(self):
        self.ui.statusMessageLabel.setText("Processing in progress")
        app.processEvents()

        Post_Threshold = self.SWATDIR + '/Post_Threshold'
        HRU_shp = self.SWATDIR + '/Watershed/Shapes/hru1.shp'
        Pre_Thresh_HRU = self.SWATDIR + '/Watershed/Grid/hrus1'
        ThreshHRU = Post_Threshold + '/Raster/thresh_HRU'

        if os.path.exists(Post_Threshold):
            shutil.rmtree(Post_Threshold)
            os.makedirs(Post_Threshold)
        else:
            print("Post_Threshold folder does not exist within SWAT model folder")

        if os.path.exists(Post_Threshold + '/Raster'):
            shutil.rmtree(Post_Threshold + '/Raster')
            os.makedirs(Post_Threshold + '/Raster')
        else:
            os.makedirs(Post_Threshold + '/Raster')
        print("Post_Threshold folder structure created within your SWAT model folder...")
```

```
        thresholdApplied = 0
        rst_array = self.readHruRaster()
        hru_list = self.readHruShape()

        #Creates a new array with the same dimensions as the HRU raster array and initializes all
value to -99
        new_hru_array = np.zeros((len(rst_array), len(rst_array[0])), dtype='a9')
        zeroIdx = np.where(new_hru_array == b'')
        new_hru_array[zeroIdx] = b'-99'

        for hruIndex, hru in enumerate(hru_list):
            idx = np.where(rst_array == int(hru))
            new_hru_array[idx] = str(hruIndex + 1).encode()

        #print("past here")
        #print("creating masks")
        import numpy.ma as ma
        fill_value = b'-99'
        new_hru_array = ma.masked_array(new_hru_array, new_hru_array == fill_value)

        from scipy.spatial import cKDTree
        x, y = np.mgrid[0:new_hru_array.shape[0], 0:new_hru_array.shape[1]]
        xygood = np.array((x[~new_hru_array.mask], y[~new_hru_array.mask])).T
        xybad = np.array((x[new_hru_array.mask], y[new_hru_array.mask])).T
        #print("ckdetree")
        new_hru_array[new_hru_array.mask] =
new_hru_array[~new_hru_array.mask][cKDTree(xygood).query(xybad)[1]]
        # print("ckdetree complete")
        ds1 = gdal.Open(self.filename)

        if ds1 is None:
            print('Could not open targeted file')
            sys.exit(1)
        targeted_array = np.array(ds1.GetRasterBand(1).ReadAsArray())

        tar_hru_array = np.array([['']*len(new_hru_array[0])]*len(new_hru_array), dtype='a8')

        tIdx = np.where(targeted_array == 1)

        for i in range(0, len(new_hru_array)):
            for j in range(0, len(new_hru_array[0])):
                if targeted_array[i][j] == 1:
                    tar_hru_array[i][j] = new_hru_array[i][j] + b"t"
                else:
                    tar_hru_array[i][j] = new_hru_array[i][j]
```

```
        # print("loop 1 ends")
        hru_range = list(set(tar_hru_array.flatten()))

        # print("loop 2 starts")
        self.hru_dict = {}
        for hru in hru_range:
            t_count = len(np.where(tar_hru_array == hru + b"t")[0])
            hru_count = len(np.where(tar_hru_array == hru)[0])
            t_hru_ratio = float(t_count) / float(t_count + hru_count)
            if np.where(tar_hru_array == hru + b"t"):
                self.hru_dict[hru.split(b"t")[0]] = float(t_hru_ratio)
        print("loop 2 ends")
        print("Processing is complete.")

        self.ui.statusMessageLabel.setText("CP simulation in progress")
        app.processEvents()
        i = 0
        os.chdir(self.SWATDIR + '/Scenarios/Default/TxtInOut')
        for files in glob.glob("*.mgt"):
            for line in fileinput.input([files], inplace=True):
                line = line.replace("0.000    | FILTERW", "0.000    | FILTERW")
                sys.stdout.write(line)
        self.ui.statusMessageLabel.setText("Processing complete. Please click on Run SWAT
Model under the processing menu for pre-CP simulation")
        app.processEvents()

#defining SWAT run for the pre-CP scenario
    def Select_Run_SWAT_Model(self):
        self.ui.statusMessageLabel.setText("SWAT model run in progress")
        app.processEvents()
        exe_str = self.SWATDIR + '/Scenarios/Default/TxtInOut/swat2012.exe'
        parent = subprocess.Popen(exe_str, stderr=subprocess.PIPE)
        time.sleep(5)
        self.ui.statusMessageLabel.setText("SWAT run complete. Please select your CP from the
Input CP Practice under the Scenario menu")
        app.processEvents()

        SWAT_output = self.SWATDIR + '/Scenarios/Default/TxtInOut'
        SWAT = open(SWAT_output+"\\output.hru", "r")
        SWATlines = SWAT.readlines()
        results = []

        #store the output for sediments
        months = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12']
        crops = ["PAST", "RNGE"]
```

```
        area_and_sed = []
        for a in range(9, len(SWATlines)):
            content = SWATlines[a]
            class MagicString(str):
                magicSplit = str.split
            s = MagicString(content)
            x = s.magicSplit()
            match = re.match(r"([a-z]+)([0-9]+)", x[0], re.I)
            if match:
                x = match.groups()
            area = x[5]
            # print(area.split('.'))
            if len(area.split('.')) == 3 and area.split('.')[0] in months and x[0] in crops:
                area = area.split('.')[2]
                # print("HERE Area")
                # print (area)
                tmp = (float("." + area), float(x[34])*self.hru_dict[bytes(x[1], 'utf-8')])
                area_and_sed.append(tmp)

        self.area_and_sed = sorted(area_and_sed, key=lambda area_and_sed: area_and_sed[0],
reverse=False)
        self.sediment = self.loss(self.area_and_sed)

        #store the output for organic phosphorus
        months = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12']
        area_and_OrgP = []
        for a in range(9, len(SWATlines)):
            content = SWATlines[a]
            class MagicString(str):
                magicSplit = str.split
            s = MagicString(content)
            x = s.magicSplit()
            match = re.match(r"([a-z]+)([0-9]+)", x[0], re.I)
            if match:
                x = match.groups()
            area = x[5]
            # print(area.split('.'))
            if len(area.split('.')) == 3 and area.split('.')[0] in months and x[0] in crops:
                area = area.split('.')[2]
                # print("HERE")
                tmp = (float("." + area), float(x[57])*self.hru_dict[bytes(x[1], 'utf-8')])
                area_and_OrgP.append(tmp)

        #store the output for sediment phosphorus
        months = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12']
        area_and_SedP = []
```

```
for a in range(9, len(SWATlines)):
    content = SWATlines[a]
    class MagicString(str):
        magicSplit = str.split
    s = MagicString(content)
    x = s.magicSplit()
    match = re.match(r"([a-z]+)([0-9]+)", x[0], re.I)
    if match:
        x = match.groups()
    area = x[5]
    # print(area.split('.'))
    if len(area.split('.')) == 3 and area.split('.')[0] in months and x[0] in crops:
        area = area.split('.')[2]
        # print("HERE")
        tmp = (float("." + area), float(x[58])*self.hru_dict[bytes(x[1], 'utf-8')])
        area_and_SedP.append(tmp)

#store the output for soluble phosphorus
months = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12']
area_and_SolP = []
for a in range(9, len(SWATlines)):
    content = SWATlines[a]
    class MagicString(str):
        magicSplit = str.split
    s = MagicString(content)
    x = s.magicSplit()
    match = re.match(r"([a-z]+)([0-9]+)", x[0], re.I)
    if match:
        x = match.groups()
    area = x[5]
    # print(area.split('.'))
    if len(area.split('.')) == 3 and area.split('.')[0] in months and x[0] in crops:
        area = area.split('.')[2]
        # print("HERE")
        tmp = (float("." + area), float(x[63])*self.hru_dict[bytes(x[1], 'utf-8')])
        area_and_SolP.append(tmp)

#calculating total phosphorus
tp = list(np.array(area_and_OrgP) + np.array(area_and_SedP) + np.array(area_and_SolP))
self.tp = sorted(tp, key=lambda tp: tp[0], reverse=False)
self.phosphorus = self.loss(self.tp)

#store the output for nitrate-nitrogen
months = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12']
area_and_NitN = []
for a in range(9, len(SWATlines)):
```

```
    content = SWATlines[a]
    class MagicString(str):
        magicSplit = str.split
    s = MagicString(content)
    x = s.magicSplit()
    match = re.match(r"([a-z]+)([0-9]+)", x[0], re.I)
    if match:
        x = match.groups()
    area = x[5]
    # print(area.split('.'))
    if len(area.split('.')) == 3 and area.split('.')[0] in months and x[0] in crops:
        area = area.split('.')[2]
        # print("HERE")
        tmp = (float("." + area), float(x[59])*self.hru_dict[bytes(x[1], 'utf-8')])
        area_and_NitN.append(tmp)

#store the output for organic nitrogen
months = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12']
area_and_OrgN = []
for a in range(9, len(SWATlines)):
    content = SWATlines[a]
    class MagicString(str):
        magicSplit = str.split
    s = MagicString(content)
    x = s.magicSplit()
    match = re.match(r"([a-z]+)([0-9]+)", x[0], re.I)
    if match:
        x = match.groups()
    area = x[5]
    # print(area.split('.'))
    if len(area.split('.')) == 3 and area.split('.')[0] in months and x[0] in crops:
        area = area.split('.')[2]
        # print("HERE")
        tmp = (float("." + area), float(x[56])*self.hru_dict[bytes(x[1], 'utf-8')])
        area_and_OrgN.append(tmp)

#store the output for lateral nitrate-nitrogen
months = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12']
area_and_NitN_Lat = []
for a in range(9, len(SWATlines)):
    content = SWATlines[a]
    class MagicString(str):
        magicSplit = str.split
    s = MagicString(content)
    x = s.magicSplit()
    match = re.match(r"([a-z]+)([0-9]+)", x[0], re.I)
```

```
            if match:
                x = match.groups()
            area = x[5]
            # print(area.split('.'))
            if len(area.split('.')) == 3 and area.split('.')[0] in months and x[0] in crops:
                area = area.split('.')[2]
                # print("HERE")
                tmp = (float("." + area), float(x[61])*self.hru_dict[bytes(x[1], 'utf-8')])
                area_and_NitN_Lat.append(tmp)

        #store the output for groundwater nitrate-nitrogen
        months = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12']
        area_and_NitN_Ground = []
        for a in range(9, len(SWATlines)):
            content = SWATlines[a]
            class MagicString(str):
                magicSplit = str.split
            s = MagicString(content)
            x = s.magicSplit()
            match = re.match(r"([a-z]+)([0-9]+)", x[0], re.I)
            if match:
                x = match.groups()
            area = x[5]
            # print(area.split('.'))
            if len(area.split('.')) == 3 and area.split('.')[0] in months and x[0] in crops:
                area = area.split('.')[2]
                # print("HERE")
                tmp = (float("." + area), float(x[62])*self.hru_dict[bytes(x[1], 'utf-8')])
                area_and_NitN_Ground.append(tmp)

        #calculating the total nitrogen
        tn = list(np.array(area_and_NitN) + np.array(area_and_OrgN) +
np.array(area_and_NitN_Lat) + np.array(area_and_NitN_Ground))
        self.tn = sorted(tn, key=lambda tn: tn[0], reverse=False)
        self.nitrogen = self.loss(self.tn)

    #defining CPs
    def Select_Inputyourcp(self):
        messagebox.showwarning("Input your CP practice", "Open your SWAT project folder and
define CP practice of interest. \n Else, select predefined irrigation land leveling CP from the
tool.")

    def Select_actionCP1(self):
        self.ui.statusMessageLabel.setText("CP simulation in progress")
        app.processEvents()
        i = 0
```

```
        os.chdir(self.SWATDIR + '/Scenarios/Default/TxtInOut')

    #Defining irrigation land leveling CP
        for files in glob.glob("*.hru"):
            for line in fileinput.input([files], inplace=1):
                if "HRU_SLP" in line:
                    new_value = float(re.match(r'[-+]?[0-9]*\.?[0-9]+', line.lstrip()).group()) * .9
                    p = re.compile(r'[-+]?[0-9]*\.?[0-9]+')
                    filterw_line = p.sub(str(new_value), line)
                    sys.stdout.write(line.replace(line, filterw_line))
                else:
                    sys.stdout.write(line)
            fileinput.close()

        for files in glob.glob("*.hru"):
            for line in fileinput.input([files], inplace=1):
                if "SLSUBBSN" in line:
                    new_value = float(re.match(r'[-+]?[0-9]*\.?[0-9]+', line.lstrip()).group()) * .1
                    p = re.compile(r'[-+]?[0-9]*\.?[0-9]+')
                    filterw_line = p.sub(str(new_value), line)
                    sys.stdout.write(line.replace(line, filterw_line))
                else:
                    sys.stdout.write(line)
            fileinput.close()

        self.ui.statusMessageLabel.setText("Selected CP simulated. Please click on Run SWAT
Model under the Scenario menu for the post-CP simulation")
        app.processEvents()

    #defining SWAT run for the post-CP scenario
    def Select_Run_SWAT_Model_2(self):
        self.ui.statusMessageLabel.setText("SWAT model run for post-CP scenario in progress")
        app.processEvents()
        exe_str = self.SWATDIR + '/Scenarios/Default/TxtInOut/swat2012.exe'
        parent = subprocess.Popen(exe_str, stderr=subprocess.PIPE)
        time.sleep(5)
        self.ui.statusMessageLabel.setText("Please click on Sediment under the Graphs menu")
        app.processEvents()

        SWAT_output = self.SWATDIR + '/Scenarios/Default/TxtInOut'
        SWAT = open(SWAT_output+"\\output.hru", "r")
        SWATlines = SWAT.readlines()

        #store the output for post-CP sediment
        months = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12']
        crops = ["RNGE", "PAST", "FRSD"]
```

```
        area_and_sed2 = []

        for a in range(9, len(SWATlines)):
            content = SWATlines[a]
            class MagicString(str):
                magicSplit = str.split
            s2 = MagicString(content)
            x2 = s2.magicSplit()
            match = re.match(r"([a-z]+)([0-9]+)", x2[0], re.I)
            if match:
                x2 = match.groups()
            area2 = x2[5]

            if len(area2.split('.')) == 3 and area2.split('.')[0] in months and x2[0] in crops:
                area2 = area2.split('.')[2]

                tmp = (float("." + area2), float(x2[34])*self.hru_dict[bytes(x2[1], 'utf-8')])
                area_and_sed2.append(tmp)

        self.area_and_sed2 = sorted(area_and_sed2, key=lambda area_and_sed2: area_and_sed2[0],
reverse=False)
        self.sediment2 = self.loss(self.area_and_sed2)

        #store the output for post-CP organic phosphorus
        months = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12']
        area_and_OrgP2 = []
        for a in range(9, len(SWATlines)):
            content = SWATlines[a]
            class MagicString(str):
                magicSplit = str.split
            s2 = MagicString(content)
            x2 = s2.magicSplit()
            match = re.match(r"([a-z]+)([0-9]+)", x2[0], re.I)
            if match:
                x2 = match.groups()
            area2 = x2[5]

            if len(area2.split('.')) == 3 and area2.split('.')[0] in months and x2[0] in crops:
                area2 = area2.split('.')[2]

                tmp = (float("." + area2), float(x2[57])*self.hru_dict[bytes(x2[1], 'utf-8')])
                area_and_OrgP2.append(tmp)

        #store the output for post-CP sediment phosphorus
        months = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12']
        area_and_SedP2 = []
```

```
        for a in range(9, len(SWATlines)):
            content = SWATlines[a]
            class MagicString(str):
                magicSplit = str.split
            s2 = MagicString(content)
            x2 = s2.magicSplit()
            match = re.match(r"([a-z]+)([0-9]+)", x2[0], re.I)
            if match:
                x2 = match.groups()
            area2 = x2[5]

            if len(area2.split('.')) == 3 and area2.split('.')[0] in months and x2[0] in crops:
                area2 = area2.split('.')[2]

                tmp = (float("." + area2), float(x2[58])*self.hru_dict[bytes(x2[1], 'utf-8')])
                area_and_SedP2.append(tmp)

        #store the output for post-CP soluble phosphorus
        months = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12']
        area_and_SolP2 = []

        for a in range(9, len(SWATlines)):
            content = SWATlines[a]
            class MagicString(str):
                magicSplit = str.split
            s2 = MagicString(content)
            x2 = s2.magicSplit()
            match = re.match(r"([a-z]+)([0-9]+)", x2[0], re.I)
            if match:
                x2 = match.groups()
            area2 = x2[5]

            if len(area2.split('.')) == 3 and area2.split('.')[0] in months and x2[0] in crops:
                area2 = area2.split('.')[2]

                tmp = (float("." + area2), float(x2[63])*self.hru_dict[bytes(x2[1], 'utf-8')])
                area_and_SolP2.append(tmp)

        #Calculating total phosphorus for post-CP scenario
        tp2 = list(np.array(area_and_OrgP2) + np.array(area_and_SedP2) +
np.array(area_and_SolP2))
        self.tp2 = sorted(tp2, key=lambda tp2: tp2[0], reverse=False)
        self.phosphorus2 = self.loss(self.tp2)

        #store the output for post-CP nitrate-nitrogen
        months = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12']
```

```
area_and_NitN2 = []
for a in range(9, len(SWATlines)):
    content = SWATlines[a]
    class MagicString(str):
        magicSplit = str.split
    s2 = MagicString(content)
    x2 = s2.magicSplit()
    match = re.match(r"([a-z]+)([0-9]+)", x2[0], re.I)
    if match:
        x2 = match.groups()
    area2 = x2[5]

    if len(area2.split('.')) == 3 and area2.split('.')[0] in months and x2[0] in crops:
        area2 = area2.split('.')[2]

        tmp = (float("." + area2), float(x2[59])*self.hru_dict[bytes(x2[1], 'utf-8')])
        area_and_NitN2.append(tmp)

#store the output for post-CP organic nitrogen
months = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12']
area_and_OrgN2 = []
for a in range(9, len(SWATlines)):
    content = SWATlines[a]
    class MagicString(str):
        magicSplit = str.split
    s2 = MagicString(content)
    x2 = s2.magicSplit()
    match = re.match(r"([a-z]+)([0-9]+)", x2[0], re.I)
    if match:
        x2 = match.groups()
    area2 = x2[5]

    if len(area2.split('.')) == 3 and area2.split('.')[0] in months and x2[0] in crops:
        area2 = area2.split('.')[2]

        tmp = (float("." + area2), float(x2[56])*self.hru_dict[bytes(x2[1], 'utf-8')])
        area_and_OrgN2.append(tmp)

#store the output for post-CP ateral nitrate-nitrogen
months = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12']
area_and_NitN_Lat2 = []
for a in range(9, len(SWATlines)):
    content = SWATlines[a]
    class MagicString(str):
        magicSplit = str.split
    s2 = MagicString(content)
```

```
            x2 = s2.magicSplit()
            match = re.match(r"([a-z]+)([0-9]+)", x2[0], re.I)
            if match:
                x2 = match.groups()
            area2 = x2[5]

            if len(area2.split('.')) == 3 and area2.split('.')[0] in months and x2[0] in crops:
                area2 = area2.split('.')[2]

                tmp = (float("." + area2), float(x2[61])*self.hru_dict[bytes(x2[1], 'utf-8')])
                area_and_NitN_Lat2.append(tmp)

        #store the output for post-CP groundwater nitrate-nitrogen
        months = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12']
        area_and_NitN_Ground2 = []
        for a in range(9, len(SWATlines)):
            content = SWATlines[a]
            class MagicString(str):
                magicSplit = str.split
            s2 = MagicString(content)
            x2 = s2.magicSplit()
            match = re.match(r"([a-z]+)([0-9]+)", x2[0], re.I)
            if match:
                x2 = match.groups()
            area2 = x2[5]

            if len(area2.split('.')) == 3 and area2.split('.')[0] in months and x2[0] in crops:
                area2 = area2.split('.')[2]

                tmp = (float("." + area2), float(x2[62])*self.hru_dict[bytes(x2[1], 'utf-8')])
                area_and_NitN_Ground2.append(tmp)

        tn2 = list(np.array(area_and_NitN2) + np.array(area_and_OrgN2) +
np.array(area_and_NitN_Lat2) + np.array(area_and_NitN_Ground2))
        self.tn2 = sorted(tn2, key=lambda tn2: tn2[0], reverse=False)
        self.nitrogen2 = self.loss(self.tn2)

    #defining graph
    def Create_Sed(self):
        self.ui.statusMessageLabel.setText("Sediment Yield Graph Displayed")
        app.processEvents()
        x = []
        y = []
        z = []
        for i in range(0, len(self.area_and_sed)):
            x.append(self.area_and_sed[i][0])
```

```
            y.append(self.sediment[i])
            z.append(self.sediment2[i])
        xaccum = list(self.accumu(x))
        yaccum = list(self.accumu(y))
        zaccum = list(self.accumu(z))

        xaccu = list(self.accumu(x))
        yaccu = list(self.accu(y))
        zaccu = list(self.accu(z))

        xy = []

        sumxy = sum(y)
        sumx = sum(x)
        areawtave_pre = sumxy/(100*sumx)
        print ('Area-weighted ave. Sediment Yield (tons): Pre-CP = ' + str(areawtave_pre))

        xz = []

        sumxz = sum(z)
        sumx = sum(x)
        areawtave_post = sumxz/(100*sumx)
        print ('Area-weighted ave. Sediment Yield (tons): Post-CP = ' + str(areawtave_post))
        self.reduction1 = ((areawtave_post - areawtave_pre)/areawtave_pre)*100

        p1, = plt.plot(xaccu, yaccu, 'bs')
        p2, = plt.plot(xaccu, zaccu, 'g^')
        plt.legend([p1, p2], ["Pre-CP", "Post-CP"], loc=2)

        plt.ylabel('Sediment Yield (tons)')
        plt.xlabel('Contributing Watershed Area (%)')

        plt.show()

    # defining graph
    def Create_SolP(self):
        self.ui.statusMessageLabel.setText("TP Graph Displayed")
        app.processEvents()
        x = []
        y = []
        z = []
        for i in range(0, len(self.tp)):
            x.append(self.tp[i][0])
            y.append(self.phosphorus[i])
            z.append(self.phosphorus2[i])
        xaccum = list(self.accumu(x))
```

```
    yaccum = list(self.accumu(y))
    zaccum = list(self.accumu(z))

    xaccu = list(self.accumu(x))
    yaccu = list(self.accu(y))
    zaccu = list(self.accu(z))

    xy = []

    sumxy = sum(y)
    sumx = sum(x)
    areawtave_pre = sumxy/(100*sumx)
    print ('Area-weighted ave. Total Phosphorus (kg): Pre-CP = ' + str(areawtave_pre))

    xz = []

    sumxz = sum(z)
    sumx = sum(x)
    areawtave_post = sumxz/(100*sumx)
    print ('Area-weighted ave. Total Phosphorus (kg): Post-CP = ' + str(areawtave_post))

    self.reduction2 = ((areawtave_post - areawtave_pre)/areawtave_pre)*100

    p1, = plt.plot(xaccu, yaccu, 'bs')
    p2, = plt.plot(xaccu, zaccu, 'g^')
    plt.legend([p1, p2], ["Pre-CP", "Post-CP"], loc=2)

    plt.ylabel('Total Phosphorus (kg)')
    plt.xlabel('Contributing Watershed Area (%)')
    plt.show()

# defining graph
def Create_NitN_Ground(self):
    self.ui.statusMessageLabel.setText("TN Graph Displayed")
    app.processEvents()
    x = []
    y = []
    z = []
    for i in range(0, len(self.tn)):
        x.append(self.tn[i][0])
        y.append(self.nitrogen[i])
        z.append(self.nitrogen2[i])
    xaccum = list(self.accumu(x))
    yaccum = list(self.accumu(y))
    zaccum = list(self.accumu(z))
```

```
    xaccu = list(self.accumu(x))
    yaccu = list(self.accu(y))
    zaccu = list(self.accu(z))

    xy = []

    sumxy = sum(y)
    sumx = sum(x)
    areawtave_pre = sumxy/(100*sumx)
    print ('Area-weighted ave. Total Nitrogen (kg): Pre-CP = ' + str(areawtave_pre))

    xz = []

    sumxz = sum(z)
    sumx = sum(x)
    areawtave_post = sumxz/(100*sumx)
    print ('Area-weighted ave. Total Nitrogen (kg): Post-CP = ' + str(areawtave_post))

    self.reduction3 = ((areawtave_post - areawtave_pre)/areawtave_pre)*100

    p1, = plt.plot(xaccu, yaccu, 'bs')
    p2, = plt.plot(xaccu, zaccu, 'g^')
    plt.legend([p1, p2], ["Pre-CP", "Post-CP"], loc=2)

    plt.ylabel('Total Nitrogen (kg)')
    plt.xlabel('Contributing Watershed Area (%)')
    plt.show()

# defining graph
def Create_Graph2(self):
    N = 1
    ind = np.arange(N) # the x locations for the groups
    offset = 0.05
    width = 0.24    # the width of the bars

    fig = plt.figure()
    ax = fig.add_subplot(111)

    baseline = [self.reduction1]
    kwargs = {"hatch":'x'}
    rects1 = ax.bar(offset+ind, baseline, width, color='w', ecolor='k', **kwargs)

    leaf = [self.reduction2]
    kwargs = {"hatch":'.'}
    rects2 = ax.bar(offset+ind+width, leaf, width, color='w', ecolor='k', **kwargs)
```

```
        ultrapeer = [self.reduction3]
        kwargs = {"hatch":'/'}
        rects3 = ax.bar(offset+ind+width+width, ultrapeer, width, color='w', ecolor='k', **kwargs)

        # add labels
        ax.set_ylabel('% Reduction in Pollutant Losses')
        ax.set_xticks(offset+ind+width+width/3)

        ax.legend( (rects1[0], rects2[0], rects3[0]), ('Sediment', 'TP', 'TN'), loc = 'best')
        plt.tick_params(
            axis='x',          # changes apply to the x-axis
            which='both',      # both major and minor ticks are affected
            bottom='off',      # ticks along the bottom edge are off
            top='off',         # ticks along the top edge are off
            labelbottom='off') # labels along the bottom edge are off
        plt.show()

    # defining cumulative function
    def accu(self, list):
        total = 0
        for x in list:
            total += x
            yield total

    # defining cumulative function as percentage
    def accumu(self, list):
        total = 0
        for x in list:
            b = sum(list)
            total += x
            yield float(float(total)/float(b))*100

    def loss(self, l):
        results = []
        for x in l:
            results.append(x[0]*x[1]*100)
        return results

    # defining help function
    def help(self):
        self.ui.statusMessageLabel.setText("Help Menu")
        app.processEvents()
        path_to_notepad = 'C:\\Program Files\\Microsoft Office\\Office15\\WINWORD.exe'
        path_to_file = 'C:\\temp\\CP Tool
Testing\\CP_Tool_Python34\\Code\\CPTool_Singh_Outline_28thJan2015.doc'
        subprocess.call([path_to_notepad, path_to_file])
```

```
if __name__ == "__main__":
    app = QtGui.QApplication(sys.argv)
    win = MainWindow()
    sys.exit(app.exec_())
```