VELAS: A FULLY-DISTRIBUTED DAILY HYDROLOGIC FEEDBACK MODEL WITH EMPHASIS ON VEGETATION, LAND COVER, AND SOIL WATER DYNAMICS

A DISSERTATION IN
Geosciences
and
Computer Science

Presented to the Faculty of the University
of Missouri – Kansas City in partial fulfillment of
the requirements for the degree

DOCTOR OF PHILOSOPHY

by

CHANGHUI PARK

B.S., Kongju National University, 2000
M.S., Kongju National University, 2002

Kansas City, Missouri
2012

VELAS: A FULLY-DISTRIBUTED DAILY HYDROLOGIC FEEDBACK MODEL WITH EMPHASIS ON VEGETATION, LAND COVER, AND SOIL WATER DYNAMICS

Changhui Park, Candidate for the Doctor of Philosophy Degree

University of Missouri – Kansas City, 2012

ABSTRACT

We present a daily hydrologic feedback model, VELAS (the VEgetation-LAnd cover-Soil water dynamics), to simulate daily responses of hydrologic processes including interception, runoff, evapotranspiration, infiltration, and recharge under various conditions of vegetation, land cover, and soil in a fully-distributed manner. The daily soil water balance is a key element to link surface and subsurface models as it calculates infiltration and groundwater recharge by considering a time delay routing through a vadose zone down to the groundwater table. MODFLOW is adopted to simulate groundwater flow and interaction with surface water components as well. The model can be localized by simple modification of soil and crop properties. The application of VELAS to a watershed in the Geum River Basin in Korea shows different daily responses of hydrologic feedbacks for different types of land cover in the same watershed. The comparison between the estimated runoff from VELAS and the observed runoff data shows a good correlation with the coefficient of 0.81. The

calculated groundwater elevation and the observed groundwater elevation also show a good correlation with the coefficient of 0.98 and the percent error less than 1.8%. The exteneded application to the entire Geum River Basin for the climate change and land cover transition during the period from 2011 to 2050 shows the capability of VELAS to predict long-term hydrologic feedbacks. The results of land cover change prediction by the Land Change Modeler show a strong agreement between the actual land cover and the predicted land cover. The predicted responses of hydrologic feedbacks reflect the impacts of the climate and land cover change.

APPROVAL PAGE

The faculty listed below, appointed by the Dean of the School of Graduate Studies, have examined a dissertation titled “VELAS: a fully-distributed daily hydrologic feedback model with emphasis on vegetation, land cover, and soil water dynamics”, presented by Changhui Park, candidate for the Doctor of Philosophy degree, and hereby certify that in their opinion it is worthy of acceptance.

Supervisory Committee

Jejung Lee, Ph.D., Committee Chair and Research Advisor
Department of Geosciences

Syed E. Hasan, Ph.D.
Department of Geosciences

Wei Ji, Ph.D.
Department of Geosciences

Yugyung Lee, Ph.D.
Department of Computer Science and Electrical Engineering

Vijay Kumar, Ph.D.
Department of Computer Science and Electrical Engineering

Jerry R. Richardson, Ph.D.
Department of Civil and Mechanical Engineering

CONTENTS

LIST OF ILLUSTRATIONS

Figure Page

LIST OF TABLES

ACKNOWLEDGEMENTS

It was a long voyage which could not have been completed without the kindhearted help of many people. First of all, I would like to sincerely thank Dr. Jejung Lee, my research advisor, for his invaluable advice, discussion, and encouragement to complete this dissertation. He was Polaris of my academic voyage. I also would like to express my gratitude to my committee members, Dr. Syed E. Hasan, Dr. Wei Ji, Dr. Yugyung Lee, Dr. Vijay Kumar, and Dr. Jerry R. Richardson, for their priceless advice, guidance, and support.

I am truly grateful to Prof. Min-Ho Koo (Kongju National University in Korea) for providing me research opportunities and his numerous advices for the groundwater modeling. I also would like to thank Mrs. Rita Messina, the former administrative assistant of Geosciences. She always helped me to solve non-academic difficulties during my coursework.

I am heartily grateful to my parents, Jusuk Park and Kyesoon Baek, and my parents-in-law, Buhyun Ryu and Gabyun Park. I will never forget their infinite support and endless love to me.

Finally, I am ever thankful to God, the Creator and the Saviour, who always strengthens me.

To my family:

my beloved wife
Jera Ryu

my dear son and daughter
Elijah and **Claire**

and

my first child in heaven
Jerome

Glory be to the Father, and to the Son, and to the Holy Spirit.
As it was in the beginning, is now, and ever shall be, world without end.
Amen.

CHAPTER 1

INTRODUCTION

1.1 Statement of Problems

The hydrological phenomena in a watershed are not independent events but rather interactive feedbacks between hydrologic processes. Comprehensive and quantitative understandings of hydrologic feedback processes are important for the effective water budget and the provision against potential risks to water resources (Winter et al., 1998; Guo, 2002). Precipitation, the major water input to a watershed, is partitioned by various hydrologic processes: interception, runoff, evapotranspiration, infiltration, and recharge (Brooks et al., 2003; Ward and Trimble, 2004; Brutsaert, 2005; Gupta, 2010). Precise partitioning of precipitated water in a watershed is important in view of providing critical information to water resource development and water use planning. Vegetation, land cover, and soil are non-hydrologic components but critical ground factors that control the hydrologic processes on the boundary between the atmosphere and the subsurface. For example, deforestation is one of the most frequently discussed disrupting factors on the responses of hydrologic feedbacks by vegetation. Loss of vegetation and forest increases runoff and stream discharge and decreases evapotranspiration. Lin and Wei (2008) investigated the impact of deforestation by timber harvesting to show an offsetting effect between deforestation and climate variability in the

Willow watershed, Canada. Coe et al. (2009) studied the historical influence of deforestation on the stream flow of the Amazon River to predict future stream discharge associated with potential deforestation and climate change. A transition of land cover is another non-hydrologic impacting factor to the change of hydrologic processes. For example, the expanding of urban area encroaches the area of vegetation or soil surface by covering with impervious surfaces and ultimately increases surface runoff, stream discharge, and flooding potential (White and Greer, 2006; Suriya and Mudgal, 2012). Especially the change of land cover from vegetated areas to urban areas results in frequent flash flooding from intensified rainfall, which has been increased due to the climate change. According to the Intergovenmental Panel on Climate Change (IPCC) report (2007), global warming will continuously progress until the year of 2100. Loaiciga et al. (1996) showed that global warming would affect local or regional hydrologic systems. Sato et al. (2007) studied precipitation change by global warming in Mongolia and discovered that severe droughts would be observed frequently. In case of the Korean Peninsula, the National Institute of Meteorological Research (NIMR) in Korea forcasted that the future climate change would be characterized by frequent occurences of extream weather with intensive precipitation and severe droughts (NIMR, 2004).

Various techniques and models have been developed to investigate the hydrologic feedback processes associated with non-hydrologic components. Integration of different hydrologic models in physical consistency is essential

for comprehensive understanding of hydrologic feedbacks in a watershed scale with accuracy. Table 1.1 shows a simple summary of exemplary integerated hydrologic models, MIKE SHE which is originated from the Système Hydrologique Européen (SHE) model (Abbott et al., 1986a, b), Soil and Water Assessment Tool (SWAT) (Arnold et al., 1993), and Water and Energy Transfer between Soil, Plants, and Atmosphere under quasi Steady State (WetSpass) (Batelaan and De Smedt, 2001). MIKE SHE is one of the most complicated hydrologic models focusing on surface and sub-suface water systems for various spatial scales and temporal resolutions (Graham and Butts, 2005). MIKE SHE has been applied for various regions and hydrological environments to estimate hydrologic processes such as groundwater recharge in a semi-arid region in Canada (Smerdon et al., 2010), surface runoff in a limited-water supply region in northwest China (Zhang et al., 2008), and large scale surface-subsurface hydrological modeling in Denmark (Henriksen et al., 2003). MIKE SHE has capabilities to provide extensive information of water resources in a watershed but needs extensive amount of input data. Lack of input data frequently limits model performance and quality of the model results from MIKE SHE (Sophocleous and Perkins, 2000; Xie and Cui, 2011). SWAT is a semi-distributed and daily watershed hydrologic model to predict the effect of land management decisions on water, sediment, nutrient and pesticide yields with reasonable accuracy in large watersheds (Neitsch et al., 2005).

Table 1.1 Summary of SHE, SWAT, and WetSpass

Feature \ Model	MIKE SHE	SWAT	WetSpass
Purpose	Synthetic simulation of suface-subsurface water and contaminents transport	prediction of the effect of management decisions on water, sediment, nutrient and pesticide yields	Estimation of the distributed long-term recharge
Integrated Hydrologic Components	interception, overland flow, ET, channel flow, Soil and ground water interaction, snow melt	interception, runoff, ET, soil zone water, recharege, base flow, snow melt	interception, runoff, ET, recharge
Spatial Discritization	distributed	semi-distributed	distributed
Temporal Scale	scalable	scalable	fixed (two season long-term)
Advantages	one of the most complicated model, surface-subsurface modeling	GIS supports, public domain model, clear watershed scale water balance, many applications	GIS supports, minimum data requirements, land cover change, rapid application
Limitations	extensive data requirements, limited accessability (commercial and protected source)	semi-distributed model, developed for watersheds in US	fixed temporal scale, designed for Belgium environments

Many studies have shown the performance and capability of SWAT to estimate hydrologic processes in various watersheds. Wu and Johnston (2007) applied SWAT to a watershed in the Ontonagon River Basin, US to estimate the

hydrologic response to climatic variability. Oeurng et al. (2011) used SWAT to estimate seiment transport and organic carbon yield in a large agricultural watershed in southwest France. SWAT has been applied frequently for surface dominated hydrologic processes such as nutrient concenturation (Pisinaras et al., 2010), stream flow simulation (Huang et al., 2009), and runoff estimation (Kannan et al., 2008). However, SWAT has limitation to simulate groundwater flow because it is a semi-distributed model based on a Hydrological Response Unit (HRU), a basic unit assumed to be homogeneous in the response of hydrologic process to the change of land cover (Kim et al., 2008; Chung et al., 2010). In view of the recent consideration of hydrological modeling that involves a non-linear and heterogeneous surface water and groundwater system, this limitation is critical (Graham and Butts, 2005). WetSpass is a relatively recent hydrologic model to estimate fully distributed and long-term average spatial patterns of interception, surface runoff, actual evapotranspiration, and groundwater recharge under quasi steady state condition (Batelaan, 2006). WetSpass has been applied for distributed recharge estimation (Batelaan and De Smedt, 2007; Dams et al., 2008; Tilahun and Merkel, 2009). One of remarkable components in WetSpass is a sub-cell composition for land cover heterogeneity. Although WetSpass simulates only two seasons, winter (dry) and summer (wet), land cover transition in a cell is involved in the calculation of hydrologic feedbacks durng simulation periods. It is inevitable that the integrated watershed models require an enormous

amount of input data to assure the accuracy of simulation results. However, WetSpass requires minimum amount of datasets that are commonly available: vegetation, land cover, soil, and weather data incluing precipitation, temperature, and wind speed. However, WetSpass is a quasi-steady state model and excludes in-depth soil water balance.

Current integrated hydrologic models have several advantage and limitations as mentioned above. The motivation of this research starts from endeavours to overcome such model limitations: difficulty of modeling for data-limited regions (MIKE SHE), limited performance for non-linear and heterogeneous surface water and groundwater interaction system model (SWAT), and limited temporal resolution of model simulation (WetSpass). Data availability and hydrologic condition of watershed may vary by region and climate. Field data are frequently limited by measurement scale and quantities (Sophocleous and Perkins, 2000). Area of interest also varies from a large watershed to a part of a single watershed such as a crop patch. Therefore, data tolerance, spatial scalability, integrating soil water balance, and consideration of land surface condition should be major priorities of a hydrologic feedback model. We developed a new model referred to as the VEgetation-LAnd cover-Soil water dynamics model (VELAS) in the present study. VELAS calculates interception, runoff, evapotranspiration, soil water variation, and recharge under various conditions of vegetation, land cover, and soil. A daily response of soil water balance dependent upon the vegetation-land cover dynamics is a

key feature of the VELAS model. The application to a watershed in the Geum River Basin in Korea demonstrates that the required input dataset of the model are commonly obtainable and simple, but the modeling results show accurate responses of hydrologic processes. The model is validated with the observed stream flow data and groundwater elevation data to show the capability of a fully synchronized simulation of hydrologic processes.

1.2 Objectives

The ultimate goal of the research is to develop a simple hydrologic feedback model for comprehensive understanding the impacts of vegetation, land cover, and soil water dynamics on the water balance. The specific objectives of the research are as follows:

- Quantifying the impacts of vegetation and land cover on hydrologic feedbacks.
- Coupling of soil water dynamics through a root zone soil.
- Predicting hydrologic feedbacks for climate change scenarios
- Construction of GIS-based hydrologic database coping with VELAS.

1.3 Literature Review

The overview of hydrologic processes shows that the spatial and temporal heterogeneity of vegetation and land cover is an important factor affecting water balance in a watershed. A crop land can be changed from bare

soil to vegetated land by following vegetation growth. Anthropogenic activities such as an expansion of urban area increase the area of impervious surface and reduce vegetated or bare soil areas. Cruise et al. (2010) showed that a large land transition from forest to agricultural land caused a decrease of stream discharge in the southeastern United States for the last 20 years. Zhang and Schilling (2006) investigated the effect of two different land covers, grass and bare soils, on the hydrologic responses in the Walnut Creek watershed in Iowa, US. They found out that the reduced groundwater recharge was observed on the grass covered area due to the lower soil moisture by the evapotranspiration process. Hence, the vegetation and land cover dynamics should be carefully considered in the calculation of soil water balance.

Interception is the first hydrologic response to the rainfall, and its amount is significant in a highly vegetated region (Muzylo et al., 2009). Hence, the estimation of interception should not be underestimated. The interception is dependent upon the ecological properties of vegetation. Leaf area index (LAI) is the most frequently used and critical property of vegetation for the interception calculation. The SWAT model (Arnold et al., 1993) and SHE model (Abbott et al., 1986a, b) calculate interception as a function of LAI. The Agricultural Policy/Environmental eXtender (APEX) model (Williams et al., 2008) calculates interception as a function of LAI and ground-above plant materials. The interception process is also highly dependent on the rainfall intensity (Aston, 1979; Ramirez and Senarath, 2000). Thus, both LAI and rainfall

should be taken into account in the interception calculation. Merriam (1960) intorduced an exponential model accounting LAI and rainfall. Aston (1979) improved the Merriam's model by introducing a fraction of rainfall intercepted by canopy. The Merriam's model has been applied to various cases by De Jong and Jetten (2007), Kozak et al. (2007), and Ragab and Bromely (2010).

Part of water after the interception flows over a land surface in a form of runoff. Frequently-used runoff estimation methods are the rational method, the curve number (CN) method, and the Green-Ampt (GA) method. The rational method is simple, easy to apply, and good to estimate average peak discharge, but is limited to small watersheds. The CN method and the GA method basically take into account the effect of soil moisture. Wilcox et al. (1990) showed that the GA method yields slightly better results than the CN method but the CN method was simple to use. King et al. (1999) performed a comparative study between CN and GA methods on the Goodwin Creek Watershed in the US in different time scales. The results from both models were very similar but GA method has limitations for simulating of seasonal variety. The advantages of simplicity, predictability, stability, and responsiveness to complex watershed characteristics make the CN method most applicable to the runoff estimation (Ponce and Hawkins, 1996). Many efforts have been made to improve the CN method for various cases of hydrologic models (Schulze, 1995; Li and Gowing, 2005; Neitsche et al., 2005;

Baltas et al., 2007; Moretti and Montanari, 2007; Kim and Lee, 2008; Hawkins et al., 2010).

Another surficial process from the rainfall is evapotranspiration. The Penman-Monteith (PM) equation is the most widely accepted evapotranspiration method. The PM equation, derived originally from the Penman equation (Penman, 1948), is a combined expression of energy, aerodynamics, and plant canopy resistance (Monteith, 1965). While the PM equation includes vegetation properties, other evapotranspiration equations such as the Hargreaves equation and the Priestley-Taylor equation do not include them. Many hydrological models adopt the PM equation to estimate evapotranspiration from the vegetated surface (Aydin et al., 2005; Chen et al., 2005; Neitsch et al., 2005; Batelaan and De Smedt, 2007; Williams et al., 2008). In the Food and Agricultural Organization (FAO) irrigation and drainage paper 56, Allen et al. (1998) introduced the reference crop evapotranspiration (ET_0) for a hypothetical crop surface based on the PM equation to provide consistent estimation of evapotranspiration in any regional and climate conditions. The FAO-PM method calculates the potential evapotranspiration (PET) for crop surface by multiplying ET_0 and a crop coefficient (K_C) representing a specific crop type. The benefits of using the FAO-PM method are its consistency of application to various crops and vegetation, simplified parameter requirement, and ease of use. The FAO-PM method has been adopted in many other

evapotranspiration studies (Eilers et al., 2007; de Silva and Rushton, 2008; Sheikh et al., 2009).

The runoff, evapotranspiration, and recharge processes are fully coupled by the soil water balance in the shallow subsurface. Castillo et al. (2003), Scipal et al. (2005), and Penna et al. (2011) studied the relationship between the soil water contents and runoff under various climate conditions. The evapotranspiration from a vegetated or bare soil cover consumes both residual water and infiltrated water in the root zone. Hence, soil water content, commonly represented as soil moisture deficit (SMD) in regards to the soil water stress against evapotranspiration, is a major controlling factor for the evapotranspiration process (Bonsu, 1997; Clark, 2002; Allen et al., 2005; Li et al., 2007). The amount of recharge is usually calculated as a residual amount of water from the soil water balance (Sophocleous, 2004). In the SHE model and the Agricultural Catchments Research Unit (ACRU) model (Schulze, 1995), a soil profile is assumed to have two layers, in which an exceeding amount of water over the maximum holding capacity of layer after budgeting becomes the amount of recharge. The use of a two-layer water balance has advantages of model simplification and ease of application. The APEX model and the SWAT model adopt a similar concept, but the soil profile is assumed to have multiple layers. However, such layer-based water balance methods are only suitable for a shallow aquifer system. The SWAT model uses an exponential decay

weighting function to calculate time delay of recharge through a vadose zone in the soil profile.

From the literature review, the selected hydrologic models are the Aston model for interception, the CN method for runoff, the FAO-PM method for evapotranspiration, and Rushton's method for soil moisture calculation. All models are integrated in VELAS in fully distributed manner and the daily transition of land cover by vegetation growth also included in the model.

CHAPTER 2

METHODOLOGY

2.1 Concept of the VELAS model

The VELAS model consists of two sub-models: surface and subsurface. The surface model involves three surface processes; interception, runoff, and evapotranspiration. The subsurface model also has three subsurface processes: soil water balance, recharge, and groundwater flow. Since soil moisture content affects runoff, evapotranspiration, and recharge, the balance of soil water is a key element to link two sub-models. The VELAS model is capable of the daily estimation of water balance in a fully distributed manner. The daily water balance of a single cell is illustrated in Figure 2.1. It is assumed that only vertical water transfer occurs in a single cell, and horizontal transferences between cells are not fully considered. Thus, the infiltration into the soil or root zone is expressed as:

$$INF_i = P_i - I_i - RO_i \tag{2.1}$$

where *INF* is the infiltration, *P* is the precipitation, *I* is the interception, *RO* is the runoff, and *i* is the time step. The soil water balance after infiltration can be expressed as Equation (2.2) in terms of the SMD:

$$SMD_i = SMD_{i-1} - INF_i + ET_i - GW_i \tag{2.2}$$

where *SMD* is the soil moisture deficit which is defined as as the depth of water required to bring the soil up to field capacity (Rushton et al., 2006), *ET* is

the evapotranspiration, and *GW* is the water supply from groundwater to the root zone. The water from the bottom of soil or root zone is calculated only when SMD_i is negative:

$$PERC_i = |SMD_i| \quad \text{for} \quad SMD_i < 0 \tag{2.3}$$

where *PERC* is the water from the bottom of the soil or root zone.

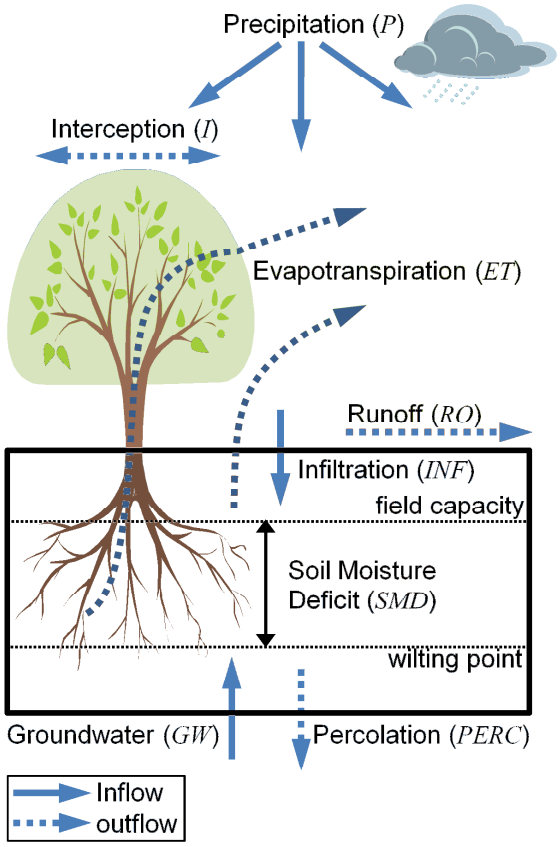

Figure 2.1 Illustration of single cell water balance.

Equations (2.1) through (2.3) play a key role of water transfer between the surface model and the subsurface model. In the case of groundwater flow, a MODFLOW model (Harbaugh et al., 2000) is fully integrated with the VELAS model. The VELAS model also takes into account the seasonal land cover transition by the growth of vegetation. Figure 2.2 shows a schematic diagram of land cover transition on a vegetated cell. A cell initially identified as bare soil is gradually changed into a vegetated cell by the vegetation growth. The fractional size of the vegetated area is zero or small during the non-growing season but becomes wider during the growing season. The land transition and the change of vegetation growth occur linearly between the growing season and non-growing season. Batelaan and De Smedt (2007) suggested a concept of sub-cell composition for the land cover heterogeneity, and VELAS adopted the same concept. Therefore, the variation of land cover fraction is one of the key considerations of the VELAS model.

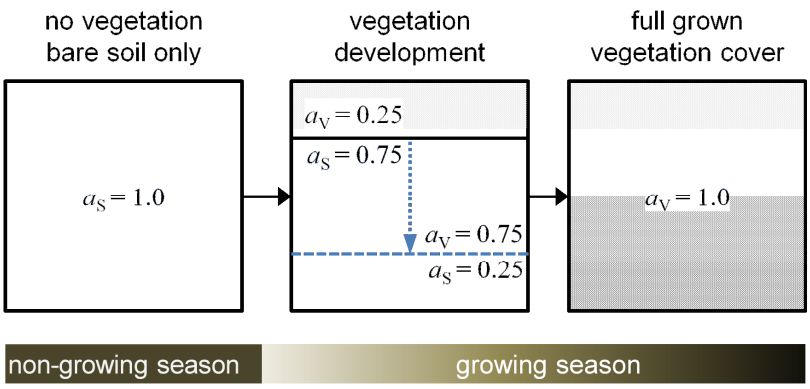

Figure 2.2 Schematic diagram of land cover transition on a vegetated cell.

2.2 Interception

On the vegetated area, water from rainfall is usually intercepted by the vegetation. Merriam (1960) introduced an exponential model to estimate rainfall interception as:

$$I = a_V \cdot SC_{max} \cdot [1 - \exp(-P / SC_{max})] \quad (2.4)$$

where I is the interception [mm], a_V is the areal fractions of vegetation [-], SC_{max} is the maximum storage capacity of canopy [mm], and P is the precipitation [mm]. Aston (1979) discussed that the Merriam model overestimated interception, and suggested a modified equation by introducing a fraction of rainfall intercepted by canopy, k [-], as:

$$I = a_V \cdot SC_{max} \cdot [1 - \exp(-kP / SC_{max})] \quad (2.5)$$

and the net amount of precipitation reaching land surface, P_{net} [mm], is calculated as:

$$P_{net} = P - I \quad (2.6)$$

The parameter k can be described as the inverse of the canopy open depending on the Leaf Area Index (LAI). De Jong and Jetten (2007) derived the relationship between k and LAI from the data of Aston (1979):

$$k = 0.065 \cdot \mathrm{LAI} \quad (2.7)$$

where LAI is the ratio of the functional leaf area of the canopy to the ground area [-] (Asner et al., 2003).

Kozak et al. (2007) compared combinations of the Merriam model (Merriam, 1960) and the van Dijk model (van Dijk and Bruijnzeel, 2001) with

two SC_{max} calculation methods (von Hoyningen-Huene, 1983; Brisson et al., 1998). They discovered that the combination of the Merriam model in Equation (2.5) with the von Hoyningen-Huene's SC_{max} calculation method, as shown in Equation (2.8) below, produces the most accurate result in comparison to the other combinations.

$$SC_{max} = 0.935 + 0.498 \cdot LAI - 0.00575 \cdot LAI^2 \quad (2.8)$$

The VELAS adopted Equation (2.8) for the calculation of SC_{max}.

2.3 Surface Runoff

Surface runoff may occur when the amount of the throughfall and the precipitation over the storage of the canopy interception exceeds the infiltration capacity of soil. However, the infiltration capacity is not the only factor. Many other factors such as rainfall intensity, soil texture, land cover, and slope affect the surface runoff calculation.

The United States Department of Agriculture, National Resources Conservation Service (NRCS) has developed a runoff model known as the NRCS Curve Number (CN) method once known as the SCS (Soil Conservation Service: the former name of NRCS) CN method, to estimate the daily surface runoff accounting for the hydraulic properties of soil, land use, slope, and soil moisture content in the root zone (USDA-NRCS, 2004). The equation describes the total storm runoff from the total storm rainfall as:

$$RO = \frac{(P_{net} - I_a)^2}{(P_{net} - I_a + S)} \quad \text{for} \quad P_{net} > I_a \tag{2.9}$$

where RO is the surface runoff [mm], I_a is the initial abstractions [mm], and S is the retention parameter depending on the soil conditions [mm]. Schulze (1995) described the initial abstractions, I_a, in terms of S by the empirical relationship as:

$$I_a = C_{I_a} \cdot S \tag{2.10}$$

where C_{I_a} is the regression coefficient depending on the vegetation, land use, and site conditions [-]. The usual value for C_{I_a} is 0.2 in many cases (USDA-NRCS, 2004). Schulze (1995) used various numbers for the regression coefficient depending on the surface conditions: 0.05 to 0.15 for the compacted soil or impervious surface and 0.3 to 0.4 for the forest or high surface roughness conditions, which mostly occur immediately after the rainfall. Woodward et al. (2003) suggested 0.05 of C_{I_a} instead of 0.2 for a better result. The retention parameter, S, varies with the CN [-].

$$S = 25.4 \cdot \left(\frac{1000}{\text{CN}} - 10 \right) \tag{2.11}$$

The NRCS introduced the Antecedent Runoff Condition (ARC) to describe the factors affecting the variation of CN values (USDA-NRCS, 2004). The ARC is classified into three antecedent runoff conditions; ARC-I for dry conditions, ARC-II for average conditions, and ARC-III for wet conditions (USDA-NRCS, 2004). The USDA-NRCS Conservation Engineering Division (1986) provides

suggested curve numbers under an average ARC and 5% of the slope (CN_{II}) for combinations of three types of areas; urban, agricultural, arid and semiarid rangeland, and four hydrologic soil groups; A, B, C, and D by the USDA-NRCS Soil Survey Division (1993). CN_{II} can be adjusted for the ARC-I and ARC-III. Tarboton (2003) used equations (2.12) and (2.13) to calculate the CN values for the ARC-I and ARC-III conditions:

$$CN_{I} = \frac{4.2 \cdot CN_{II}}{10 - 0.058 \cdot CN_{II}} \tag{2.12}$$

$$CN_{III} = \frac{23 \cdot CN_{II}}{10 + 0.13 \cdot CN_{II}} \tag{2.13}$$

where CN_{I}, CN_{II}, and CN_{III} are the curve numbers for the ARC-I, ARC-II, and ARC-III conditions respectively.

Neitsche et al. (2005) presented a new equation to make the retention parameter change with soil characteristics and soil moisture conditions.

$$S = S_{I} \cdot \left(1 - \frac{SW}{SW + \exp(w_1 - w_2 \cdot SW)} \right) \tag{2.14}$$

where S_{I} is the maximum value of the retention parameter using CN_{I} [mm], SW is the soil water content of the entire profile excluding the amount of water at wilting point [mm], and w_1 and w_2 are the shape coefficients [-]. The shape coefficients are determined by solving Equation (2.14).

$$w_1 = \ln\left(\frac{\theta_{FC}}{1 - S_{III} \cdot S_{I}^{-1}} - \theta_{FC} \right) + w_2 \cdot \theta_{FC} \tag{2.15}$$

$$w_2 = \frac{\ln\left(\frac{\theta_{FC}}{1 - S_{III} \cdot S_I^{-1}} - \theta_{FC}\right) - \ln\left(\frac{\theta_{SAT}}{1 - 2.54 \cdot S_I^{-1}} - \theta_{SAT}\right)}{\theta_{SAT} - \theta_{FC}} \quad (2.16)$$

under the following assumptions:

$$S = S_I \quad \text{for} \quad SW = \theta_{WP} \quad (2.17)$$

$$S = S_{III} \quad \text{for} \quad SW = \theta_{FC} \quad (2.18)$$

$$S = 2.54 \quad \text{for} \quad SW = \theta_{SAT} \quad (2.19)$$

where θ_{WP}, θ_{FC}, and θ_{SAT} are the moisture contents at the wilting point, the field capacity, and the saturation [mm], respectively. S_{III} is the retention parameter using CN_{III} [mm].

Basically, CN_{II} for the ARC-II is determined under the assumption that the land surface slope is 5%. Thus, CN_{II} can be adjusted to a different slope (Neitsche et al., 2005):

$$CN_{IIs} = \frac{CN_{III} - CN_{II}}{3}\left[1 - 2 \cdot \exp(-13.86 \cdot slp)\right] + CN_{II} \quad (2.20)$$

where CN_{IIs} is the slope-adjusted curve number for the ARC-II and *slp* is the surface slope.

2.4 Soil Water Balance and Evapotranspiration

Evapotranspiration is controlled by climate and soil moisture conditions. When a water supply from soil is unlimited and continued, evapotranspiration occurs at the full potential rate depending on the climate conditions. However, if a water supply from soil is limited, evapotranspiration

may occur at the reduced potential rate, called actual evapotranspiration, depending on soil properties and crop types. Rushton (2003) introduced the concept of soil moisture deficit to calculate the soil moisture storage in the soil profile and actual evapotranspiration. The soil moisture deficit (SMD) is defined as the depth of water required to bring the soil up to field capacity (Rushton et al., 2006). When the soil moisture content reaches to field capacity, the SMD is zero.

Allen et al. (1998) suggested the Penman-Monteith equation to estimate the reference evaportranspiration (ET_0) from the reference surface, which is covered by a hypothetical grass reference crop with an assumed crop height of 0.12 m, a fixed surface resistance of 70 s/m and an albedo of 0.23. The reference evapotranspiration is calculated as,

$$ET_0 = \frac{0.408 \cdot \Delta(R_n - G) + \gamma \frac{900}{T+273} u_2 (e_s - e_a)}{\Delta + \gamma(1 + 0.34 \cdot u_2)} \quad (2.21)$$

where R_n is the net radiation at the crop surface [MJ/m^2/day], G is the soil heat flux density [MJ/m^2/day], T is the mean daily air temperature [^{o}C], u_2 is the wind speed at 2 m height [m/s], e_s is the saturation vapour pressure [kPa], e_a is the actual vapour pressure [kPa], Δ is the slope of the vapour pressure curve [kPa/^{o}C], and γ is the psychrometric constant [kPa/^{o}C]. The potential evapotranspiration of the specific crop, PTR [mm], can be calculate from the reference evapotranspiration of grass,

$$PTR = K_C \cdot ET_0 \quad (2.22)$$

where K_C is the crop coefficient of the specific crop [-] (Allen et al., 1998) and ET_0 is the reference evaportranspiration [mm]. The potential evaporation from bare soil also can be calculated from the reference evapotranspiration,

$$PEV = K_E \cdot ET_0 \tag{2.23}$$

where PEV is the potential evaporation from bare soil [mm] and K_E is the evaporation coefficient [-], which is 1.05 for semi-arid climate and 1.10 for temperate climate (Rushton et al, 2006). PTR and PEV may be changed by the SMD variation and the soil and crop properties.

Rushton et al. (2006) introduced three stages in the evaporation process using the total evaporable water (TEW) and the readily evaporable water (REW). Stage I in Figure 2.3a represents the evaporation that occurs at the potential rate when the SMD is zero or small. Stage II represents the reduced evaporation by insufficient soil water supply when the SMD is between REW and TEW. Stage III represents the ceased evaporation due to the depletion of soil water. The transitions from stage I to stage II and from stage II and III are identified by REW and TEW, respectively. Thus, the potential evaporation can be adjusted to the actual evaporation as follows,

$$AEV = K'_S \cdot PEV \tag{2.24}$$

where AEV is the actual evaporation [mm] and K'_S is the evaporation stress coefficient [-]. K'_S can be calculated as,

$$K'_S = \frac{TEW - SMD}{TEW - REW} \quad \text{when } REW < SMD < TEW \tag{2.25}$$

where TEW is the total evaporable water [mm], SMD is the soil moisture deficit [mm], and REW is the readily evaporable water [mm] and TEW is calculated with,

$$\mathrm{TEW} = (\theta_{\mathrm{FC}} - 0.5 \cdot \theta_{\mathrm{WP}}) \cdot Z_{\mathrm{E}} \cdot 1000 \tag{2.26}$$

where Z_E is the maximum evaporation depth [m]. REW can be acquired from field observation.

The actual transpiration can be estimated in the same manner with the calculation of the actual evaporation. Figure 2.3b shows the transpiration by the soil moisture. In this case, REW and TEW are altered to PAW (Plant Available Water) and $\mathrm{PAW}_{\max}$ (Maximum Plant Available Water), respectively. Thus, the actual transpiration is calculated as,

$$\mathrm{ATR} = K_{\mathrm{S}} \cdot \mathrm{PTR} \tag{2.27}$$

where ATR is the actual transpiration [mm] and K_S is the soil stress coefficient [-]. K_S can be calculated as

$$K_{\mathrm{S}} = \frac{\mathrm{PAW}_{\max} - \mathrm{SMD}}{\mathrm{PAW}_{\max} - \mathrm{PAW}} \quad \text{when } \mathrm{PAW} < \mathrm{SMD} < \mathrm{PAW}_{\max} \tag{2.28}$$

where $\mathrm{PAW}_{\max}$ is the maximum plant available water [mm] and PAW is the plant available water [mm]. TAW is calculated with,

$$\mathrm{PAW}_{\max} = (\theta_{\mathrm{FC}} - \theta_{\mathrm{WP}}) \cdot Z_r \cdot 1000 \tag{2.29}$$

where Z_r is the root depth [m]. PAW can be obtained from $\mathrm{PAW}_{\max}$ as,

$$\mathrm{PAW} = p \cdot \mathrm{PAW}_{\max} \tag{2.30}$$

where p is the average depletion fraction of PAW_{max} [-]. The value of p varies by the crop type (Allen et al., 1998).

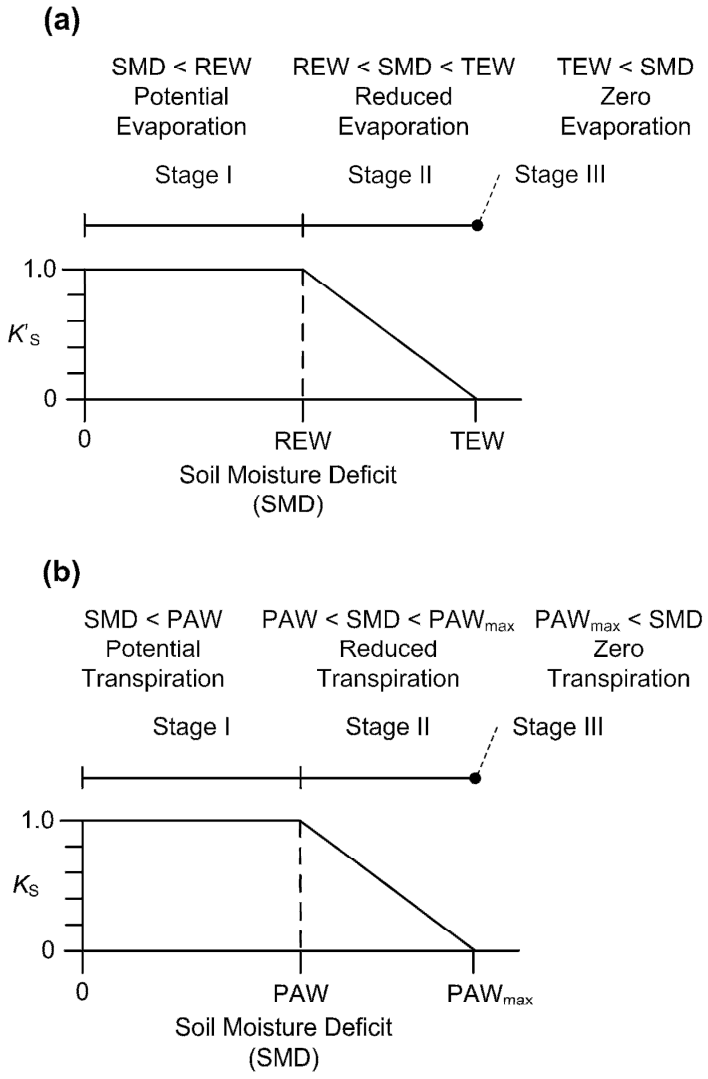

Figure 2.3 Evaporation and Transpiration change by SMD variation: (a) Evaporation, (b) Transpiration. Modified from Rushton et al. (2006)

Transpiration and evaporation can be combined by weighting areal fractions of vegetation and soil.

$$K_{CE} = a_V \cdot K_C + a_S \cdot K_E \tag{2.31}$$

$$\mathrm{RAW} = a_V \cdot \mathrm{PAW} + a_S \cdot \mathrm{REW} \tag{2.32}$$

$$\mathrm{TAW} = a_V \cdot \mathrm{PAW}_{max} + a_S \cdot \mathrm{TEW} \tag{2.33}$$

where K_{CE} is the combined coefficient for evapotranspiration [-], a_V and a_S are the areal fractions of vegetation and soil [-], RAW is the readily available water [mm], and TAW is the total available water [mm]. Thus, the potential evapotranspiration can be estimated from Equations (2.22), (2.23), and (2.31).

$$\mathrm{PET} = K_{CE} \cdot \mathrm{ET}_0 \tag{2.34}$$

where PET is the potential evapotranspiration [mm]. The actual evapotranspiration is also calculated from Equations (2.32), (2.33), and (2.34).

$$\mathrm{AET} = K_r \cdot \mathrm{PET} \tag{2.35}$$

$$K_r = \frac{\mathrm{TAW} - \mathrm{SMD}}{\mathrm{TAW} - \mathrm{RAW}} \quad \text{when RAW} < \mathrm{SMD} < \mathrm{TAW} \tag{2.36}$$

where AET is the actual evapotranspiration [mm] and K_r is the soil water stress coefficient [-].

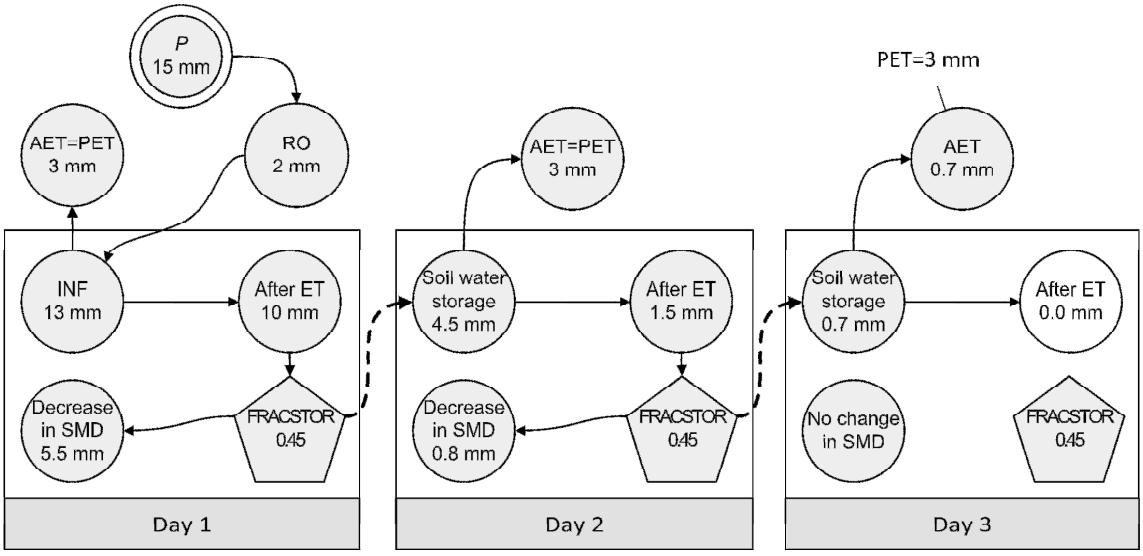

Figure 2.4 Example of soil water balance calculation. Modified from Rushton et al. (2006)

Soil water storage near the surface can be estimated with an empirical factor FRACSTOR representing a storage retention fraction (Rushton, 2003; Rushton et al., 2006). Figure 2.4 is a diagram for daily water balance change in the near surface soil water storage as an example. On Day One, 13 mm of infiltration occured from 15 mm of precipitation. Evapotranspiration occurs at the potential rate of 3 mm because the infiltration is sufficient. After the evapotranspiration process, the soil water storage at the end of Day One is 4.5 mm (0.45 times 10) and the remaining water is 5.5 mm which reduces the SMD. On Day Two, evapotranspiration still occurs at the potential rate of 3 mm because the soil water storage from Day One is 4.5 mm. Using the same calculation at the end of Day One, the soil water storage at the end of Day Two

is 0.7 mm and the remaining water is 0.8 mm which again reduces the SMD. On Day Three, evapotranspiration occurs at the reduced rate of 0.7 mm due to the depletion of the soil moisture and the SMD is retained.

2.5 Recharge Estimation

Water from the bottom of soil or root zone is infiltrated through the vadose zone and eventually reaches the groundwater table. Neitsch et al. (2005) suggested an equation to estimate the actual recharge to the aquifer considering the time lag by the vadose zone routing.

$$RCH_i = PERC_i \cdot [1 - \exp(-1/\delta_{gw})] + RCH_{i-1} \cdot \exp(-1/\delta_{gw}) \qquad (2.37)$$

where *RCH* is the recharge [mm], *PERC* is the water from the bottom of the soil or root zone [mm], δ_{gw} is the delay time of the overlying geologic formations [day], and *i* is the time step. Since it is unable to measure δ_{gw} directly, it can be estimated by finding the optimum value of δ_{gw} that shows the maximum correlation between the simulated recharge and the observed groundwater level (Neitsch et al., 2005; Chung et al., 2010).

CHAPTER 3

APPLICATION

3.1 Algorithm and Implementation

Figure 3.1 illustrates an algorithm for the VELAS model calculation. The first year cycle of the model is dedicated as a dummy year to determine an initial value of SMD for each cell (Rushton et al., 2006). The SMD of the last day of the dummy year, which comes from the initial run, is the initial value of SMD of the model. Each daily simulation loop includes six steps of calculation. The interception is calculated as a first step, then the runoff and the potential evapotranspiration calculations are followed. The soil water balance is calculated at the fourth, and the potential evapotranspiration is recalculated for the actual evapotranspiration at the fifth step. The recharge calculation is the last step of calculation. If the groundwater model exists, the output recharge of the model is converted into the input of RCH package in MODFLOW (Harbaugh et al., 2000) and the MODFLOW execution is called. After running MODFLOW, simulated groundwater levels are converted as an input of the VELAS model. VELAS is coded with an object-oriented language, Python. Python is suitable for processing an immense amount of spatial data using a supporting module of ArcGIS (ESRI, 2008). The VELAS main module consists of interfacing, input-output (I/O) processing, scene generating, and hydrologic calculation (Figure 3.2).

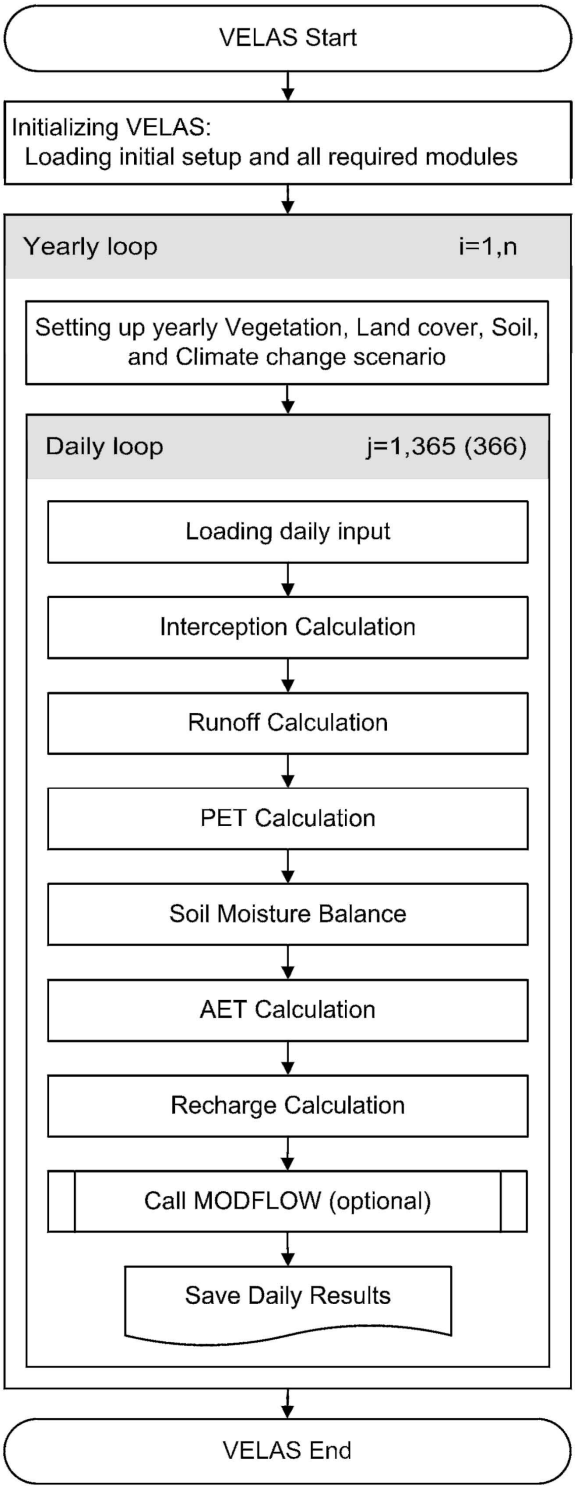

Figure 3.1 VELAS algorithm flowchart.

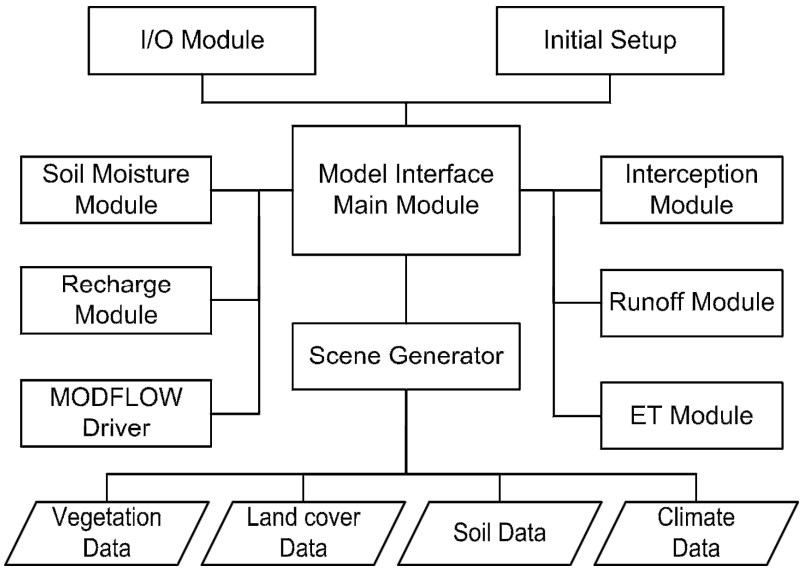

Figure 3.2 Schematic diagram of coded module structures of VELAS.

The model interface is interconnected with the main module that drives and controls all sub modules. Once the main module is invoked, variables, defined in the initial setup, are loaded at the same time. Then I/O module reads the input files. The scene generator module produces a daily near-surface soil and surface vegetation change scenario based on land cover, soil, and vegetation data for one year. The MODFLOW driver performs data transferring between the VELAS model and the MODFLOW.

3.2 Study Site

The Geum River Basin (or Geumgang River Basin) is located in the southwestern part of Korea. The basin has 14 sub-watersheds based on the

topography of the Geum River and its distribution of tributaries. VELAS was applied to one of the 14 sub-watersheds numbered as WS3010. This WS3010 watershed is located in the middle of the basin with a latitude of 36°27'00"N to 36°36'06"N and a longitude of 127°19'40"E to 127°28'15"E (Figure 3.3a). The area of WS3010 is 129.6 km^2, and the midstream of the Geum River meanders from the south to the west side of the watershed. Cultivable areas have been developed mostly around two major tributaries of the Geum River. The average elevation of the watershed is 96 m above mean sea level and its range is from 18 m to 322 m (Figure 3.3b). The slope of the watershed gradually decreases from 13.4% to 0% and its average is 3.9% (Figure 3.3c). The watershed has a temperate climate condition with a distinct rainy summer season from June to September. The annual average temperature is 11 to 12.5°C and the annual average precipitation is 1,100 to 1,300 mm/year.

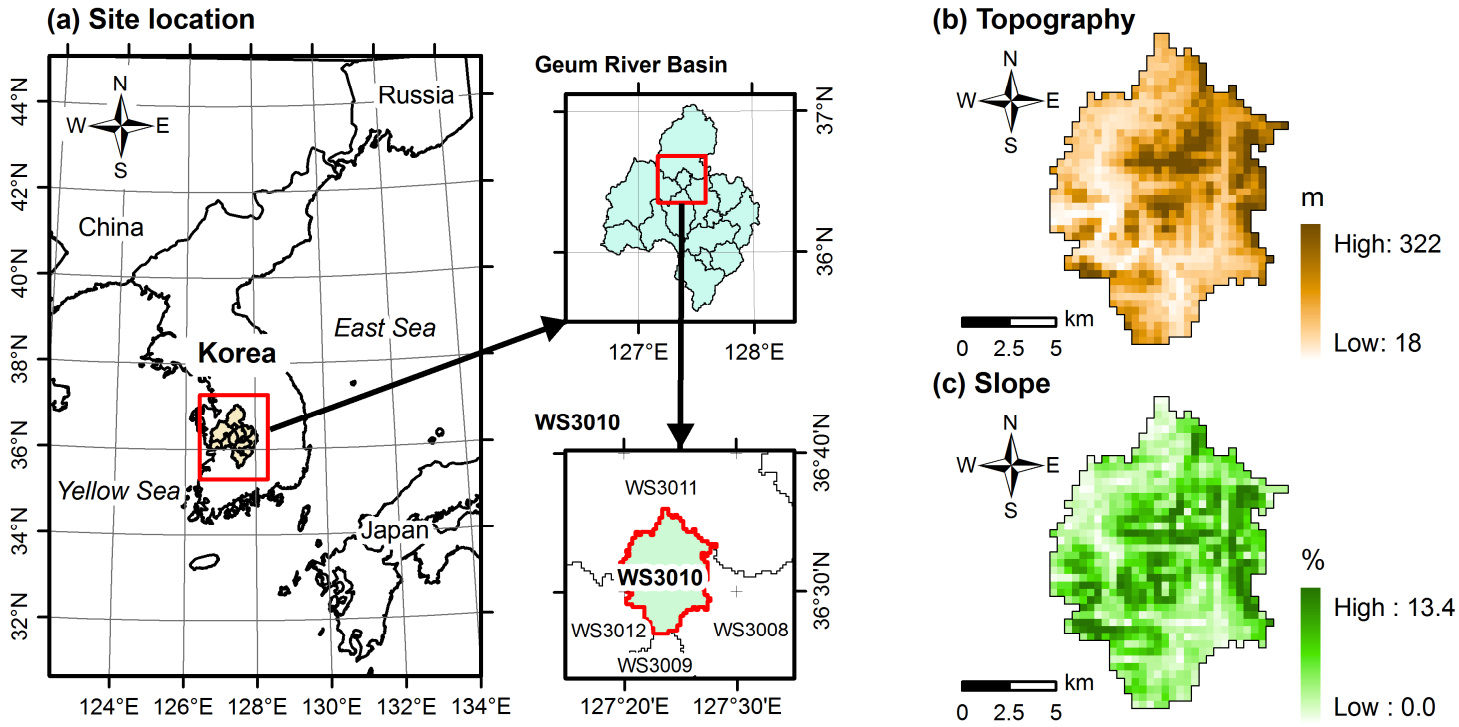

Figure 3.3 Location, topography, and slope map of study site: (a) Location, (b) Topography, (c) Slope.

3.3 Data Preparation

The watershed is treated as a raster grid. To fulfill this spatial concept, all input and output follow ESRI's ASCII raster format with the same dimensional properties including cell size, number of cells, and extent. Required datasets for the model are listed in Table 3.1.

Table 3.1 Required input datasets for the VELAS model

Dataset	unit	comments
Vegetation	-	user defined (Table 2)
Land cover	-	user defined (Table 3)
Soil	-	NRCS Soil Texture (Table 4)
Topography	m	SRTM V4.1 (Jarvis et al., 2008)
Slope	%	
Temperature	°C	daily average, min, max
Relative Humidity	%	daily average
Wind Speed	m/s	at 2 m above ground surface

The vegetation map was obtained from the Water Management Information System (WAMIS) of the Ministry of Land, Transport, and Maritime Affaires (MLTM) in Korea (Figure 3.4a). The attribute table for the vegetation map contains vegetation growth-associated parameters such as crop coefficient, height, root depth, LAI, and growing area fraction change (Table 3.2). The land cover map is based on the digital land cover map for the year of 2000 provided by the WAMIS (Figure 3.4b). The land cover map is linked to the attribute table with index numbers. The attribute table of land cover map defines curve

number and area fraction among vegetation, soil, and others for each index number (Table 3.3). The curve number definition for each land cover type basically follows the NRCS Technical Release 55 (TR-55) (USDA-NRCS Conservation Engineering Division, 1986), but new items can be added for undefined land cover types. Because rice paddy fields broadly cover the study site and are not listed on TR-55, curve numbers for the rice paddy fields from Kang et al. (2006) and Im et al. (2007) were added to the attribute table.

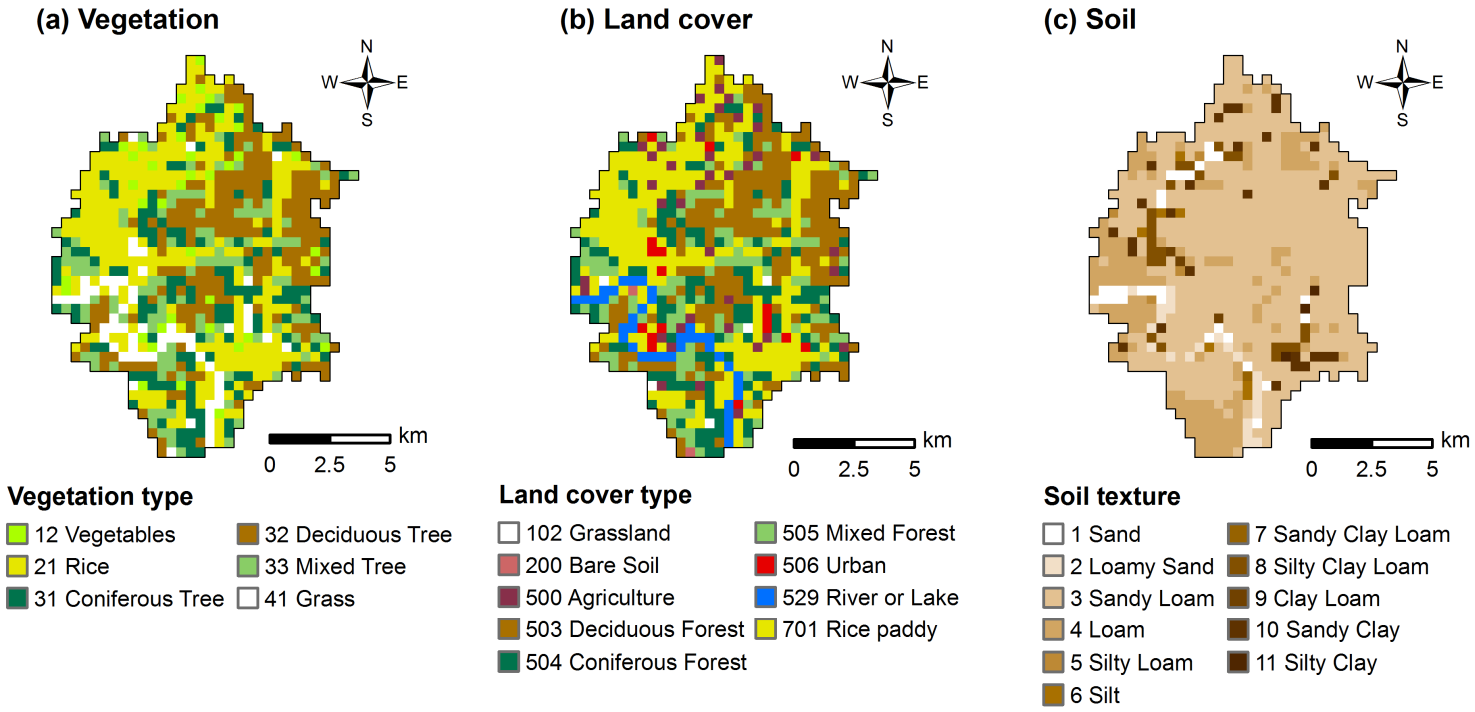

Figure 3.4 Vegetation, land cover, and soil map of study site: (a) Vegetation, (b) Land cover, (c) Soil.

Table 3.2 Vegetation properties (L_{ini}, L_{dev}, L_{mid}, L_{late}: Length of vegetation growth stages [day]; K_C: Crop Coefficient [-], Ht_{max}: Maximum height of vegetation [m]; Zr_{max}: Maximum root depth of vegetation [m]; p: average depletion fraction of PAW_{max} [-]; a_{Vmin}, a_{Vmax}: % Minimum and Maximum area of vegetation [-]; LAI_{min}, LAI_{max}; Minimum and Maximum value of leaf area index [-])

index	Vegetation	Type	Growth Stage[a]					K_C[a]			Ht_{max}[a]	Zr_{max}[a]	p[a]	Area Fraction[b]		LAI[c]	
			L_{ini}	L_{dev}	L_{mid}	L_{late}	Plant Date	K_{Cini}	K_{Cmid}	K_{Cend}				a_{Vmin}	a_{Vmax}	LAI_{min}	LAI_{max}
12	Vegetables	Crop	25	40	95	20	100	0.57	1.05	0.84	0.35	0.50	0.45	0.0	0.8	0.0	4.0
21	Rice	Crop	30	30	60	30	120	1.05	1.20	0.90	1.00	1.00	0.20	0.0	0.8	0.0	5.0
31	Coniferous Tree	C_Tree	20	10	150	30	110	1.00	1.00	1.00	15.00	2.00	0.70	0.9	1.0	7.0	8.0
32	Deciduous Tree	D_Tree	20	10	150	30	110	0.50	1.10	0.65	15.00	2.00	0.50	0.2	1.0	0.5	7.0
33	Mixed Tree	M_Tree	20	10	150	30	110	0.75	1.05	0.83	15.00	2.00	0.60	0.5	1.0	4.0	7.5
41	Grass	Grass	10	20	200	15	60	0.80	0.85	0.85	0.10	0.75	0.50	0.0	1.0	0.0	2.0

[a]Allen et al. (1998).

[b]Batelaan (2006).

[c]Kite (2002) and Zhou et al. (2006).

A soil map, which is obtained from the Korea Institute of Geoscience and Mineral Resources (KIGAM), follows the NRCS soil classification scheme (USDA-NRCS Soil Survey Division, 1993) that identifies 12 soil types by texture (Figure 3.4c). Other soil parameters related to hydraulic properties are also listed on the attribute table (Table 3.4).

Table 3.3 Land cover properties ($index_{NG}$: index number for non-growing season [-]; A, B, C, D: Hydrologic Soil Group [-]; a_{VS}: % area of vegetation and soil surface [-]; a_{IW}: % area of impervious and water surface [-])

index	Land cover	Type	$index_{NG}$	Curve Number by HSG[a]				Area Fraction[a]	
				A	B	C	D	a_{VS}	a_{IW}
102	Grassland	Crop	200	39	61	74	80	1.00	0.00
200	Bare Soil	Bare	200	77	86	91	94	1.00	0.00
500	Agriculture	Crop	200	59	74	82	86	1.00	0.00
503	Deciduous Forest	Forest	300	36	60	73	79	1.00	0.00
504	Coniferous Forest	Forest	300	40	66	77	85	1.00	0.00
505	Mixed Forest	Forest	300	38	63	75	82	1.00	0.00
506	Urban	Impervious	200	89	92	94	95	0.15	0.85
529	River or Lake	Water	529	100	100	100	100	0.00	1.00
701	Rice Paddy	Paddy	200	82[b]	82[b]	82[b]	82[b]	1.00	0.00

[a]USDA-NRCS Conservation Engineering Division (1986).

[b]Kang et al. (2006) and Im et al. (2007)

Table 3.4 Soil properties (HSG: Hydrologic Soil Group [-]; SAT: Saturation point [-]; FC: Field capacity [-]; WP: Wilting point [-]; REW: Readily available water [mm]; TEW: Total evaporable water [mm]; Z_E: Evaporation depth [m]; TENSHT: Tension height [m])

index	Soil Texture	HSG[a]	SAT[b]	FC[b]	WP[b]	REW[c]	TEW	FRAC STOR[d]	Z_E[e]	TEN SHT[f]
1	Sand	A	0.47	0.09	0.04	6.2	8.4	0.10	0.125	0.07
2	Loamy Sand	A	0.45	0.14	0.08	7.4	12.3	0.25	0.125	0.09
3	Sandy Loam	A	0.45	0.18	0.08	8.8	17.3	0.40	0.125	0.15
4	Loam	B	0.46	0.28	0.14	9.6	26.4	0.55	0.125	0.11
5	Silty Loam	B	0.48	0.31	0.11	9.2	31.3	0.60	0.125	0.21
6	Silt	C	0.48	0.31	0.06	8.4	35.6	0.65	0.125	0.61
7	Sandy Clay Loam	C	0.45	0.31	0.19	10.0	26.9	0.65	0.125	0.28
8	Silty Clay Loam	D	0.51	0.38	0.22	10.8	34.3	0.68	0.125	0.33
9	Clay Loam	D	0.47	0.35	0.21	11.0	30.6	0.70	0.125	0.26
10	Sandy Clay	D	0.44	0.36	0.25	11.0	29.4	0.70	0.125	0.29
11	Silty Clay	D	0.53	0.41	0.27	11.6	34.6	0.75	0.125	0.34
12	Clay	D	0.51	0.45	0.34	7.5	35.1	0.85	0.125	0.37

[a]USDA-NRCS Conservation Engineering Division (1986).
[b]USDA-NRCS Soil Survey Division (1993) and Saxton and Rawls (2006).
[c]Ritchie et al. (1972).
[d]Rushton (2003) and Rushton et al. (2006).
[e]Allen et al. (2005).
[f]Rawls et al. (1992).

The topography map is extracted from the Processed SRTM Data Version 4.1 (Jarvis et al., 2008) provided by the National Aeronautics and Space Administration (NASA) (Figure 3.3b). A slope map, which is required to adjust curve numbers, was created from the topography data (Figure 3.3c). Daily

weather data for the year of 2007 including precipitation, temperature, relative humidity, and wind speed were obtained from 67 monitoring stations, which cover the entirety of South Korea including the Geum River Basin, under management of the Korea Meteorological Administration (KMA). All weather data were interpolated using kriging (Davis, 2002). The long-term stream flow data was collected from the WAMIS for the model validation purpose. The stream flow data includes daily runoff and baseflow for the watershed. All datasets for the model input were preprocessed to have a size of 400 m by 400 m for a unit cell with 32 columns and 42 rows of grid totaled 810 effective cells. The groundwater level data from the National Groundwater Information Management and Service Center (GIMS) in Korea were used for the verification of recharge and groundwater simulation.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Hydrologic Responses on a Single cell with a Different Land cover

Figures 4.1 to 4.3 show the daily hydrologic responses for the year of 2007 on a single cell identified as agricultural land, forest, urban, grass land, and bare soil surfaces. The soil type of all cells is sandy loam except for the bare soil cell identified as sand. The soil properties are defined in Table 3.4. Each figure shows the responses of hydrologic components to the precipitation events, of which high peaks occurred during the wet season from late June to early September.

Figures 4.1a and 4.1b show the hydrological responses for the same agricultural land but having different vegetation on each cell; vegetable and rice paddy, respectively. In this study it is assumed that both lands are rainfed, and the planting dates are the 100^{th} and 120^{th} day of the year, respectively. The interception occurred only during the growing season when the vegetation existed, and both vegetable and rice paddy showed similar patterns of frequency and amounts of interception. The simulated runoff, however, showed distinctive patterns by the seasons, soil moisture contents, and vegetations. Since the land transition is based on the sub-cell concept (Batelaan and De Smedt, 2007), a vegetated land cover cell has a different base curve number for the growing and non-growing seasons as defined in Table 3.3

The curve numbers then change linearly between the seasons. In addition to this seasonal change, the adjustment of the curve number is made for the change of the daily soil moisture during the runoff calculation. Various base curve numbers for different vegetations are also considered. Thus, the simulated daily runoff varies for different vegetations even though the amount of precipitation is the same. A comparison between two runoff events on the 63rd and 175th day in Figure 4.1a is an example of the seasonal difference. The amount of precipitation on the 63rd day was 30.6 mm (Ⓐ) and the amount of runoff was 8.8 mm (❶). On the 175th day, however, the amount of runoff decreased to 3.2 mm (❷) despite the slight increase in the amount of precipitation, 35.6 mm (Ⓑ). It is because the base curve number of the agriculture-vegetable cell is 77 for the non-growing season and 59 for the growing season. Another example of the runoff difference is illustrated in Figure 4.1b. The amounts of precipitation on the 39th and 88th day were the same amount of 14.8 mm (Ⓒ and Ⓓ), but the runoff occurred only on the 88th day with the amount of 2.5 mm (❸) in spite of the same curve number of 77 for the non-growing season. This difference of runoff is due to the soil moisture content represented as the SMD in the model. The amounts of the SMD on the 39th and 88th day were 15.6 mm (❹) and 0.0 mm (❺), respectively. Therefore, the higher soil moisture content on the 88th day resulted in the higher runoff on the same day. The annual amounts of precipitation on both the agriculture-vegetable and the agriculture-rice paddy cells were 1,615.8

mm/year and 1,623.7 mm/year, respectively. Due to the higher base curve number of rice compared to vegetable, the greater amount of runoff occurred on the agriculture-rice paddy cell with the annual rate of 507.3 mm/year which is over two folds of the amount of 203.0 mm/year on the agriculture-vegetable cell. The variations of the SMD also affect the AET. When the SMD is lower than the RAW due to the continuous or heavy precipitation events, the cell retains enough water to be consumed by evapotranspiration with no stress, and the AET occurs at the potential rate. The reduced amounts of evapotranspiration due to the higher SMD compared to RAW are illustrated as solid white on the evapotranspiration plots in Figure 4.1. The highest SMD values for both cells occurred on the 171st day (❺and ❻), but the amount of AET on the agriculture-vegetable cell was 11% of the PET (❻) which is lower than 35% on the agriculture-rice paddy cell (❼). The difference of PET is due to the different rooting depths of vegetable and rice in the TAW calculation. Deeper rooting depth of rice results in higher TAW and relaxed soil water stress. The RAW and TAW usually change linearly with the growing and withering of natural vegetation. In case of the crop vegetation, the RAW and TAW decrease abruptly in a day due to the harvesting of crops (❼and ❽). The negative SMD values indicate that the amount of water in the soil profile exceeds field capacity, and the percolation from the bottom of the soil profile occurs with the equivalent amount of the negative SMD values. After the percolation process, water from the bottom of the soil profile flows through

the vadose zone and the aquifer recharge occurs with the time delay as explained with Equation (2.37). However, the time delay (δ_{gw}) was disregarded by the result of the simulated recharge and the observed groundwater correlation process showing the best correlation with the coefficient of 0.6 when δ_{gw} is equal to zero.

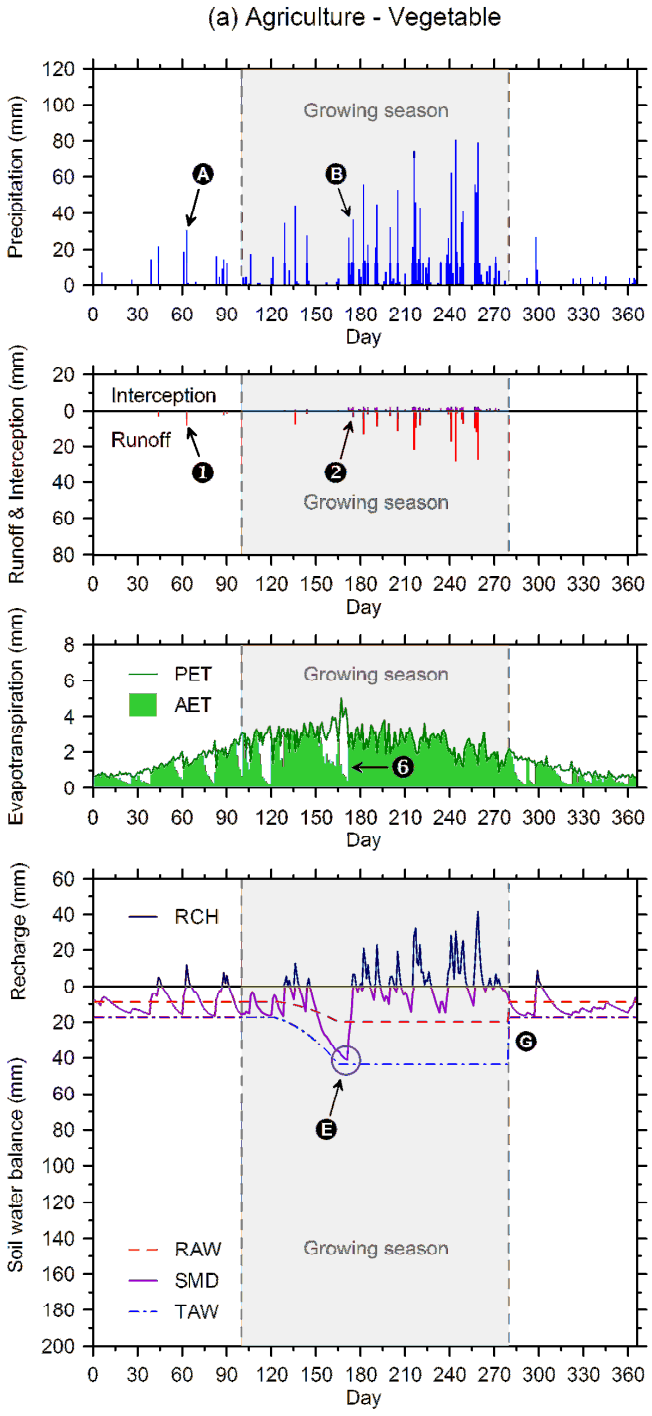

Figure 4.1 Hydrologic responses on two different crop covered agricultural land in daily time step for a year: (a) Agriculture-Vegetable, (b) Agriculture-Rice paddy.

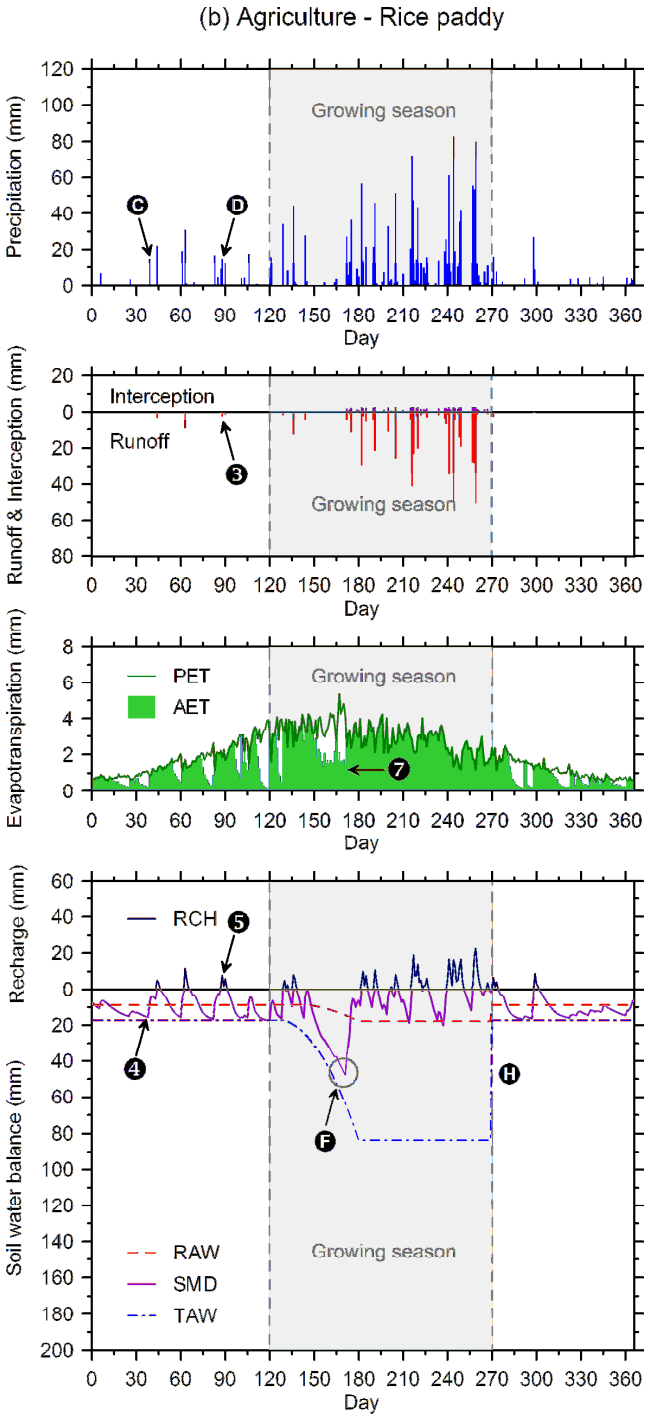

Figure 4.1 Continued.

Figure 4.2 compares the hydrological responses of the forest cell and urban cell having high contrast of land use and land cover. The forest was assumed as a mixed forest consisting of 50% coniferous and 50% deciduous trees. The deciduous trees have a foliaged period as a growing season. The high rate of interception is resulted from the high LAI values of trees while the low runoff occurred due to the small curve numbers of forest (Ⓐ). The extensive coverage of impervious surface on the urban cell limited the interception close to zero and increased runoff (Ⓑ). The AET on the forest cell reached the PET throughout the year due to the high level of soil water content (Ⓒ). The calculated AET on the urban cell reached the PET as well only when a precipitation event occurred (Ⓓ). The evapotranspiration on the urban cell is calculated as the sum of evaporation from the impervious fraction, which is only available on rainy days, and evapotranspiration from the vegetated fraction. The recharge on the forest cell occurred during the wet season (Ⓔ) while it occurs frequently throughout the year on the urban cell (Ⓕ) because the higher SMD on the forest cell requires a greater rate of infiltration to reach the field capacity.

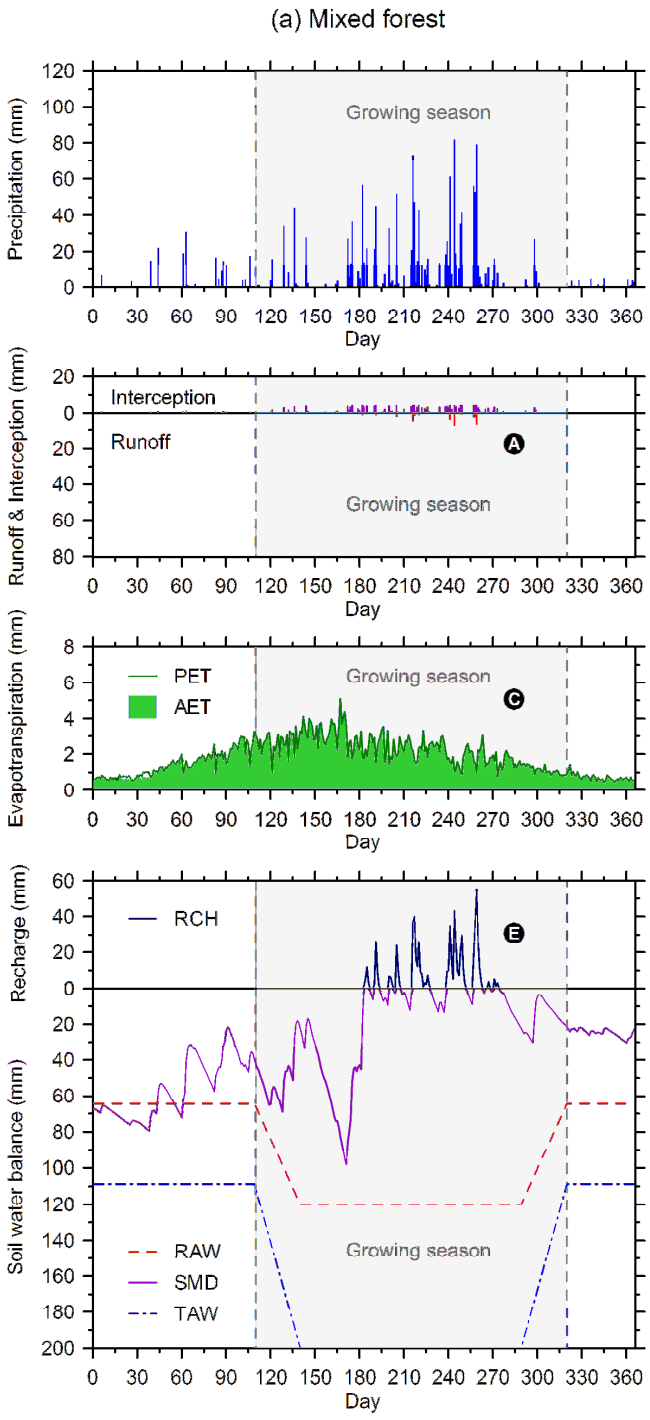

Figure 4.2 Hydrologic responses on a forest cell and an urban cell in daily time step for a year: (a) Forest, (b) Urban.

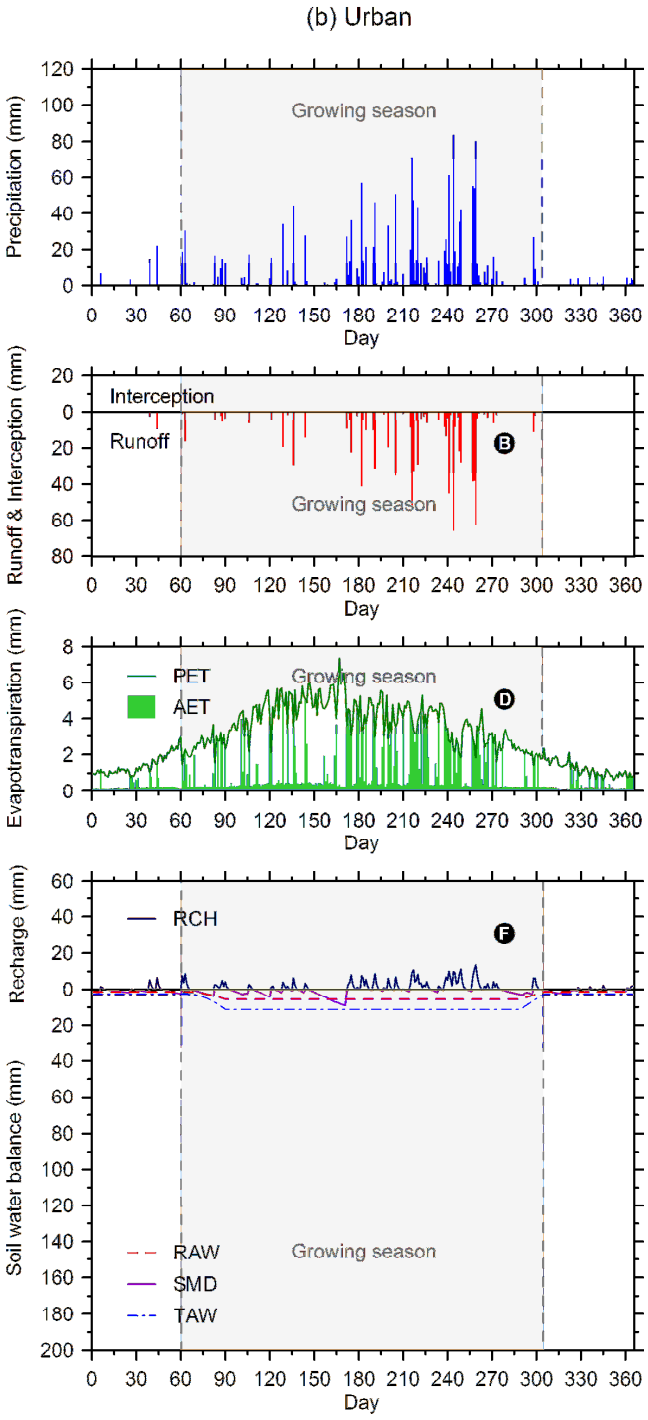

Figure 4.2 Continued.

The results of the hydrologic responses on a grassland cell and a bare soil cell are illustrated in Figure 4.3. It is assumed that the grass cell is naturally vegetated, and both cells are maintained throughout the year without any anthropogenic activities. The effects of vegetation growth on the grass cell show similar patterns with other vegetated cells explained in Figures 4.1a and 4.2a. The bare soil cell, however, shows different patterns of evapotranspiration and soil water balance. As the daily REW and TEW values are fixed throughout the year, only soil evaporation occurs. The AEV was zero for many days (Ⓐ and downward arrows) because the SMD reached the TAW and no more evaporable water was available in the soil profile on the same days of no evaporation (Ⓑ and upward arrows).

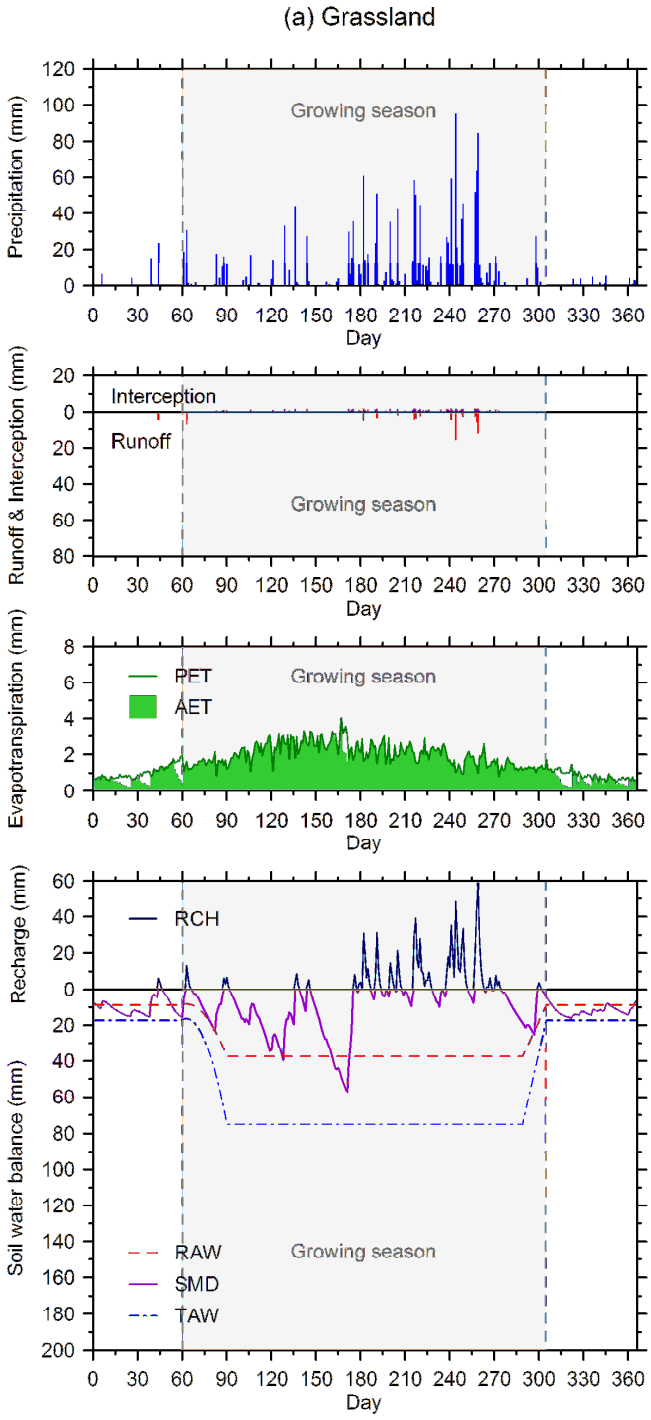

Figure 4.3 Hydrologic responses on a grassland cell and a bare soil cell in daily time step for a year: (a) Grassland, (b) Bare soil – Sand.

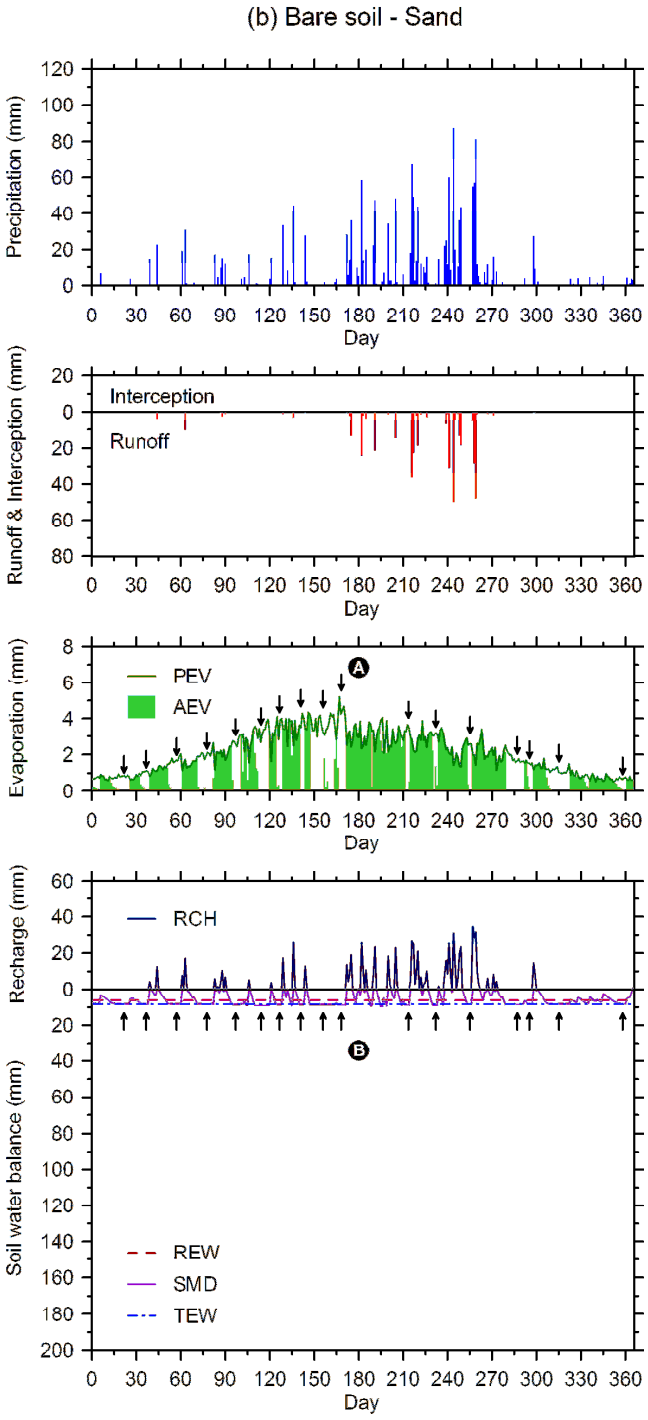

Figure 4.3 Continued.

4.2 Distributed Form of Hydrologic Responses

VELAS has the capability to calculate daily evapotranspiration, soil water balance, interception, and infiltration that leads to the daily estimation of runoff and recharge in a distributed form of space. Figure 4.4 shows the distributed hydrologic responses for seven days from the 171st to the 177th day (June 20th to 26th) in 2007. In this application, rainfall is assumed as the only source of precipitation in the watershed. The 171st day was assumed to have a dry condition without rainfall resulting in zero runoff and zero interception. SMD was the largest for the given period of comparison still providing an input of the remaining moisture for the process of AET. A serial event of rainfall was observed from the 172nd day to the 175th day (Figure 4.4a) causing the increase of interception (Figure 4.4b) and runoff (Figure 4.4c) throughout the watershed. The higher interception occurred in the forest areas having high LAI values. The agriculture and grass areas have relatively low interception while the urban, water, and bare area show the lowest interception due to the lack of vegetation. The change in runoff is affected by the precipitation, soil moisture contents, soil hydraulic properties, land cover types, and topographic slopes. The high values of curve number from low permeability of land cover in the urban and agriculture-rice paddy areas produce a large amount of runoff. It can be noted that the highest runoff is observed along the river cells (Figure 3.4b) due to the high curve number of 100 for the water surfaces in the model. In the case of the forest area, a low rate of runoff is estimated due to the low

curve number. The interception and runoff are only observed on rainy days due to Equation (2.5) and (2.9), respectively. The distributed patterns of AET are illustrated in Figure 4.4d. On the 171st day, the AET occurs at the rate of PET mainly in the forest and river areas, while other areas showed the reduced amounts of AET. The peak levels of evapotranspiration are observed on the 177th day, five days after the first rainfall event on the 172nd day. It is because the AET is limited by the levels of SMD represented as the coefficients of soil water stress (Figure 4.4e). As shown in the continuous rainfall events from the 172nd day to the 175th day, the SMD decreased continuously throughout the watershed (Figure 4.4f), and the soil water stress is relaxed. After the rainfall events, the recharge occurs in the areas where the SMD reaches below zero (Figure 4.4g). The SMD changes in the vegetated portion of the urban area occur faster than other areas due to the relatively low TAW making the area more sensitive to the change of soil moisture.

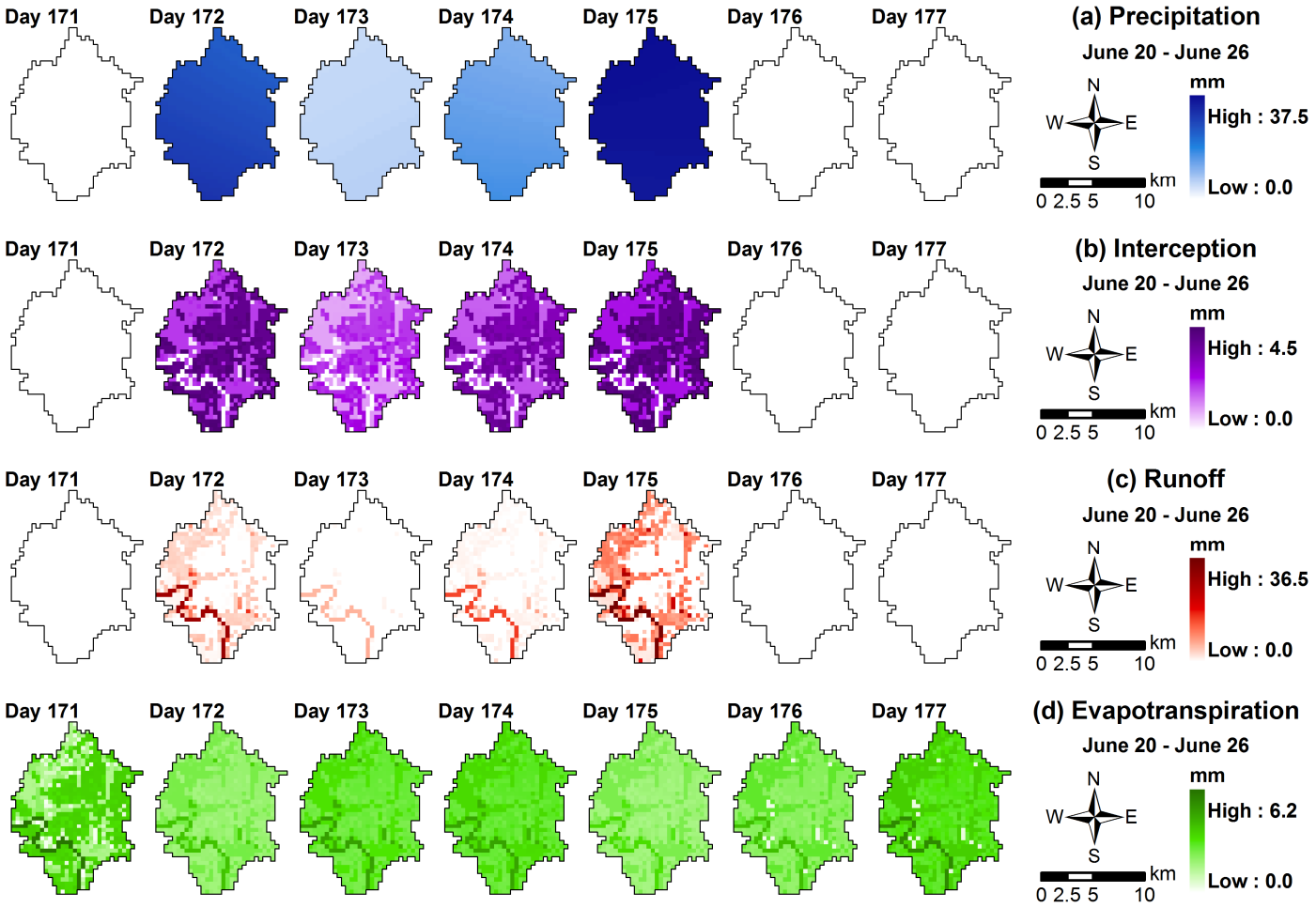

Figure 4.4 Distributed hydrologic responses for seven days from the 171st to the 177th day (June 20th to 26th) in 2007: (a) Precipitation, (b) Interception, (c) Runoff, (d) Evapotranspiration, (e) Soil water stress, (f) Soil moisture deficit, (g) Recharge.

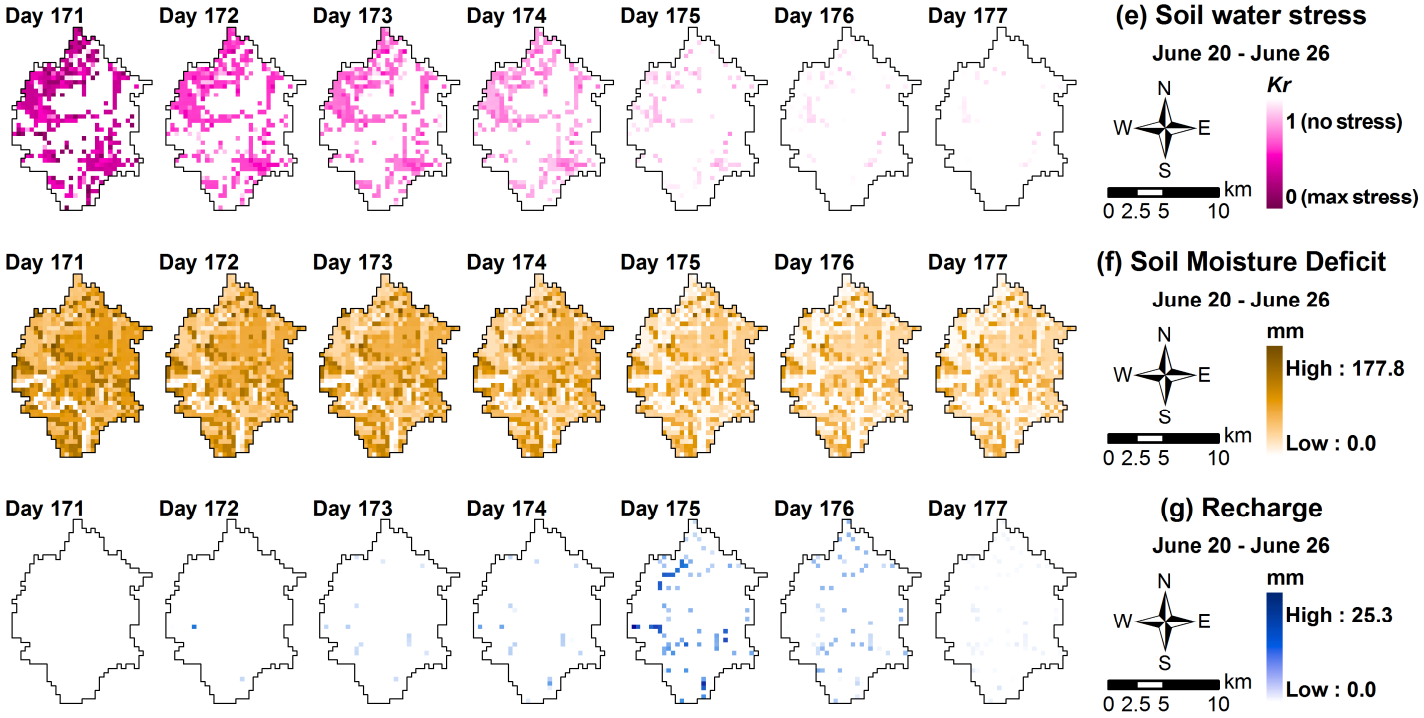

Figure 4.4 Continued.

4.3 Model Validation Using Stream Flow Data

The calculated runoff is a result of the soil moisture variation, climate change, and land cover transition with the change of vegetation growth. Thus, the comparison between the calculated runoff and observed runoff is a necessary process to validate the accuracy of the model performance. The observed runoff can be obtained from the observed stream flow data in the watershed. Since the observed stream flow includes both direct runoff and baseflow, a separation process is required. A recursive digital filter method (Eckhardt, 2005) was adopted to separate the direct runoff from the observed discharge data. Eckhardt's recursive digital filter method requires two parameters, the filter parameter and BFI_{max}, to calculate baseflow:

$$b_k = \frac{(1 - BFI_{max}) \cdot a \cdot b_{k-1} + (1 - a) BFI_{max} \cdot y_k}{1 - a \cdot BFI_{max}} \tag{4.1}$$

where b_k is the baseflow at time step k, BFI_{max} is the maximum value of baseflow index, a is the filter parameter, and y_k is the total stream flow at time step k. The suggested filter parameter, a, is 0.925 (Nathan and McMahon, 1990) and BFI_{max} for perennial stream with porous aquifer is 0.80 (Eckhardt, 2005). Figure 4.5a shows a result of cross-correlation between the observed runoff from the baseflow separation and the simulated runoff in 2007. The highest correlation is observed at zero lag with the coefficient of 0.81 as shown in Figure 4.5b.

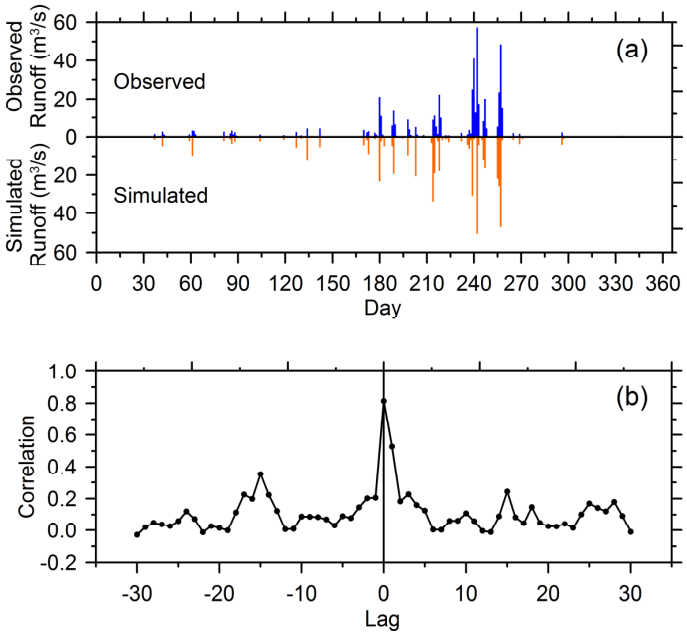

Figure 4.5 Model validation results.

4.4 Response of Groundwater to the Estimated Recharge

The accuracy of recharge estimation was verified by comparing the observed groundwater elevations with the calculated groundwater elevation from the embedded MODFLOW simulator. The site was discretized into a grid set of 32 columns and 42 rows composed by 400 m by 400 m unit cells in a single layer of an unconfined aquifer. The dimension of the discretized groundwater model is the same as the VELAS input for a simultaneous run without any scaling. The groundwater model was constructed as a transient model having 365 stress periods. Thus, both the VELAS model and the groundwater model were fully synchronized. The aquifer consists mainly of alluvium, fractured granite, and schist (Figure 4.6a). Hydraulic conductivities

for granite and alluvium are 0.2 m/day and 4.98 ~ 5.48 m/day, respectively (Domenico and Schwartz, 1990; Fetter, 2001). The specific yield for aquifer media was calibrated in a range from 0.01 to 0.2. A river package for the Geum River and drain package for tributaries were applied to the hydrologic interactions with the aquifer. Since most sections of tributaries are very narrow and almost intermittent, applying the drain package is reasonable for the given site. Effects of pumping, located in the urban and agriculture areas along the tributaries, were also included in the model. The daily use of groundwater for the watershed was 199.3 m^3/day/km^2 in 2007.

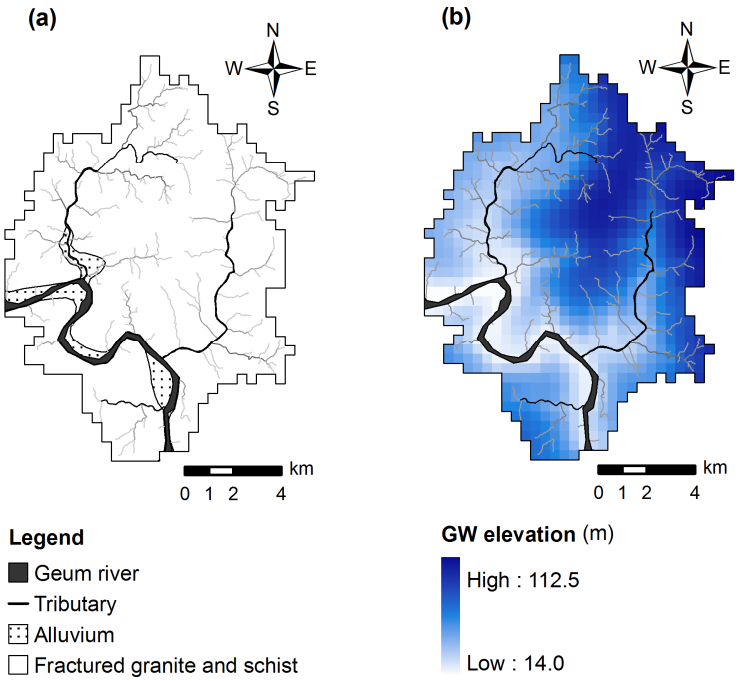

Figure 4.6 MODFLOW simulation of WS3010: (a) Geology and stream distribution, (b) Simulated groundwater elevation on the 174th day.

Figure 4.6b is the distribution of groundwater elevation on the 174th day. The groundwater elevation generally follows the topographic elevation and converges to the Geum River and tributaries. Figure 4.7a shows a daily change of the groundwater elevation between the observed and the calculated. The percent error of the groundwater estimation ranges from 0.0% to 1.8% (Figure 4.7b). The calculated groundwater elevation shows the best correlation with the observed groundwater elevation at zero lag with the coefficient of 0.98 (Figure 4.7c). The high correlation at zero lag indicates that the estimated recharge is well reflected on the calculation of the groundwater.

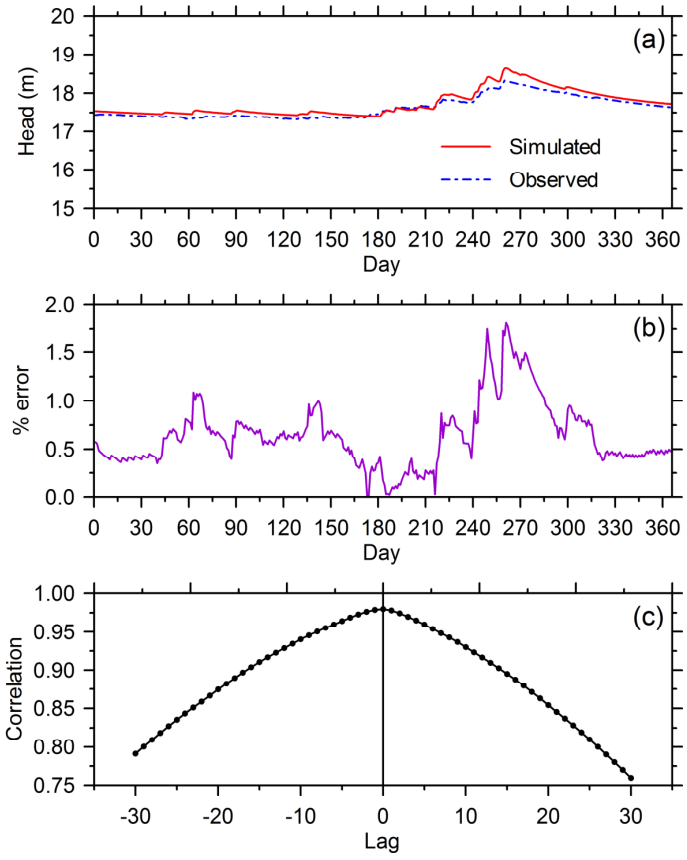

Figure 4.7 Groundwater responses to the estimated recharge.

CHAPTER 5

EXTENDED APPLICATION OF VELAS

5.1 Predictive Modeling with Climate Change and Land Cover Transition Models

The VELAS model has a capability to predict long-term hydrologic feedbacks with the impacts of climate change and land cover dynamics. The spatial scalability and the optimized data requirement of VELAS provide high degree of flexibility of application. The range of study area was extended to the entire Geum River Basin, the total area of 9,914 km^2 including all of 14 watersheds (Figure 3.3a). The modeling period was set from the year of 2001 to 2050. The actual measurements of climate change were used for the modeling period from 2001 to 2010 and the predicted climate data were assigned for the period from 2011 to 2050. To take into account the land change dynamics, a land change modeling method using an artificial neural network was introduced. The predicted land cover maps were created every 5-year interval through the modeling period. The vegetation parameters were determined from the land cover change results. Other variables including soil, slope, and topography, were assumed as fixed variables (Figure 5.1).

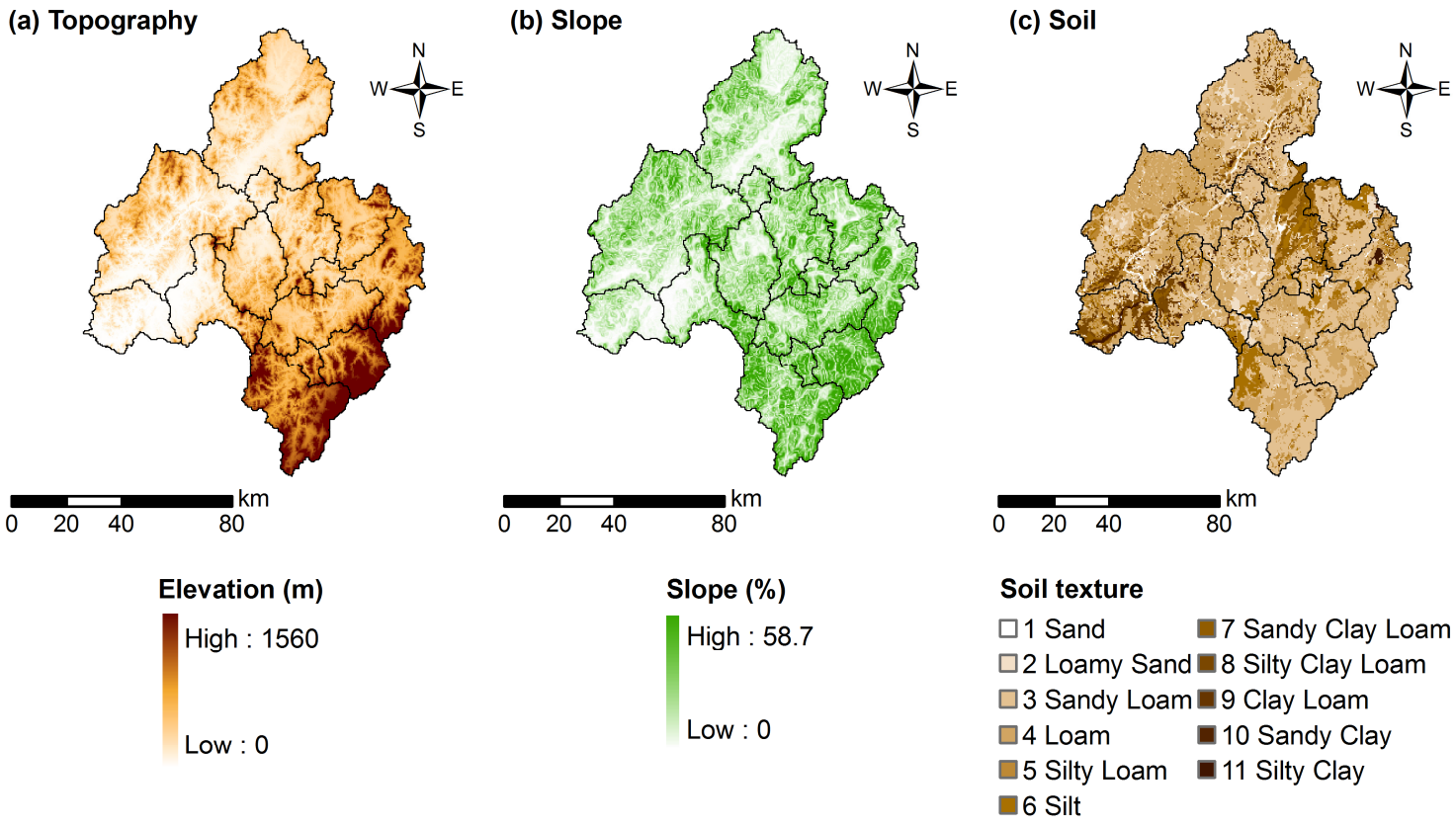

Figure 5.1 Topography, Slope, and Soil map of the Geum River Basin: (a) Topography, (b) Slope, (c) Soil.

5.2 Emission Scenarios and Climate Change Prediction Data

Climate is the most critical driving force of the hydrologic system of the Earth. The global emission of greenhouse gas into the atmosphere is considered to increase the atmospheric temperature significantly and affect the hydrologic system. The Intergovernmental Panel on Climate Change (IPCC) published the Special Report on Emission Scenarios (SRES) describing the future greenhouse gas emission (IPCC, 2000). The emission scenarios are classified into four families, A1, A2, B1, and B2, by population increase, economic growth, and technology of energy use. Figure 5.2 is a schematic diagram of four emission scenarios.

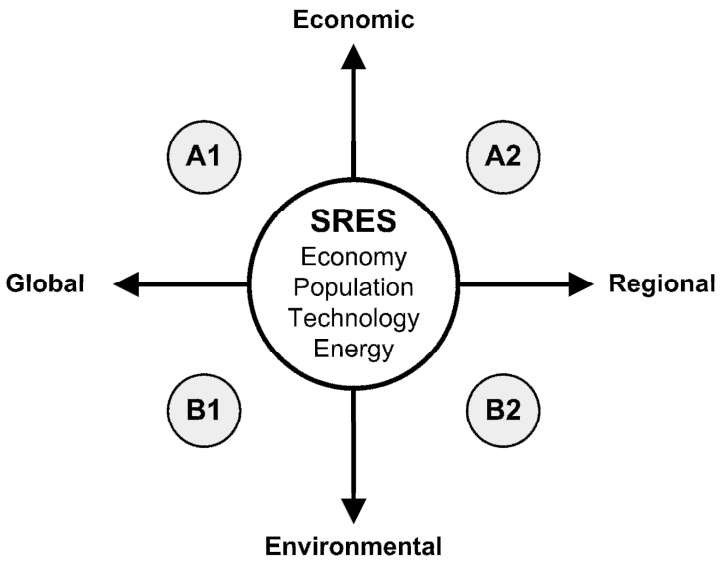

Figure 5.2 Greenhouse gas emission scenarios.

The A1 scenario assumes a rapid economic growth, peak population on 2050, and rapid introduction of technology. The A1 scenario has three subsets by energy use: A1FI (fossil energy intensive), A1T (non-fossil energy), and A1B (balanced energy use). The A2 scenario assumes that the future world is very heterogeneous in economics and technology but population increases continuously. The B1 scenario has similar condition from the A1 scenario in terms of economic and population growth, but considers environment-friendly world. The B2 scenario is characterized by an intermediate growth of economy and population. IPCC forecasted that global warming will continuously progress until the year of 2100 based on the SRES (IPCC, 2007). The National Institute of Meteorological Research (NIMR) in Korea provides climate change prediction dataset that is simulated under the A1B scenario (NIMR, 2004). The NIMR applied A1B scenario to an atmosphere-ocean coupled climate model, ECHO-G (Legutke and Voss, 1999), to simulate global scale prediction. The results from A1B scenario then were downscaled by the NIMR with Penn State/National Center for Atmospheric Research Mesoscale Model 5 (MM5) (Grell et al., 1995) to produce high-resolution datasets for the Korean Peninsula, where its latitude is from 32°58'5"N to 43°25'34"N and the longitude is from 122°56'6"E to 131°26'49"E, with 27 km of unit grid interval. Downscaled climatic elements are precipitation, temperature, and relative humidity in distributed daily basis format. Although the downscaled datasets from the NIMR have relatively higher resolution (27 km grid interval) than the results of the ECHO-

G simulation (400 km grid interval), the grid interval was still rough for the study site. In this research, kriging (Davis, 2002) was applied to the raw datasets to transform their grid size to 400 m interval. Figure 5.3 shows the predicted annual change of precipitation, temperature, and precipitation from the year of 2011 to 2050. The variation of predicted precipitation shows a decreasing trend until the year of 2027 with the average rate of 6.4 mm/year and an increasing trend after the late 2020's with the average rate of 8.9 mm/year. The predicted temperature and relative humidity show a trend of gradual increase overall with the average rate of 0.03°C/year and 1.5%, respectively. These predicted climate data are input datasets for the VELAS model.

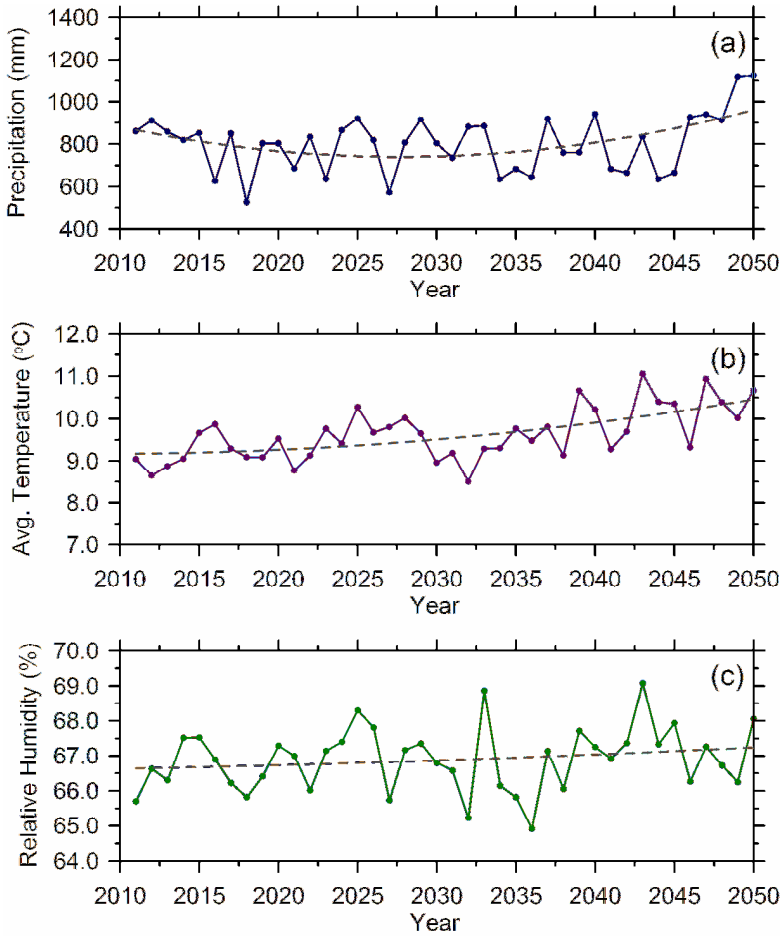

Figure 5.3 Predicted climate variables from 2011 to 2050: (a) Precipitation, (b) Average temperature, (c) Relative Humidity.

5.3 Modeling Land Cover Dynamics

The Land Change Modeler (LCM) by Clark Labs (2009) is adopted to simulate land cover transitions of the basin for the modeling period. The LCM consists of five sequential steps (Figure 5.4): Step 1) identifying major transitions, Step 2) selecting explanatory variables, Step 3) calculating transition potentials, Step 4) predicting future land cover, and Step 5)

validating results. Major transitions among land cover classes are identified from the analysis of the past transitions in Step 1. The identified major transitions are then transferred to Step 2. The explanatory variables regarded as driving forces of land transition are selected from candidate variables, which seem to have high relation with land cover classes, after evaluating their explanatory potential for each land cover class. The LCM adopts general multi-layer perceptrons (MLP) neural network (Basheer and Hajmeer, 2000) as a tool for the calculation of the transition potentials which are results of the relation between the major land cover transitions from Step 1 and the explanatory variables from Step 2. Based on the calculated transition potentials from Step 3, the LCM predicts a future land cover by using Markov Chain analysis (Eastman, 2009). The simulated results of land cover transition are validated at the last step with the Kappa indices (Pontius, 2000) and the Relative Operating Characteristic (ROC) analysis (Pontius and Schneider, 2001).

The analysis of the past land transition between two land cover maps created for the year of 1985 and 1995 makes it possible to identify major land transitions. All possible transitions among land cover classes were quantified firstly and minor transitions showing the change of area less than 20 km^2 were excluded for the enhancement of modeling efficiency. The selected major transitions are summarized in Table 5.1. The transition of Barren to Urban was selected due to its clear change trend near urban area even though the transition area is less than 20 km^2.

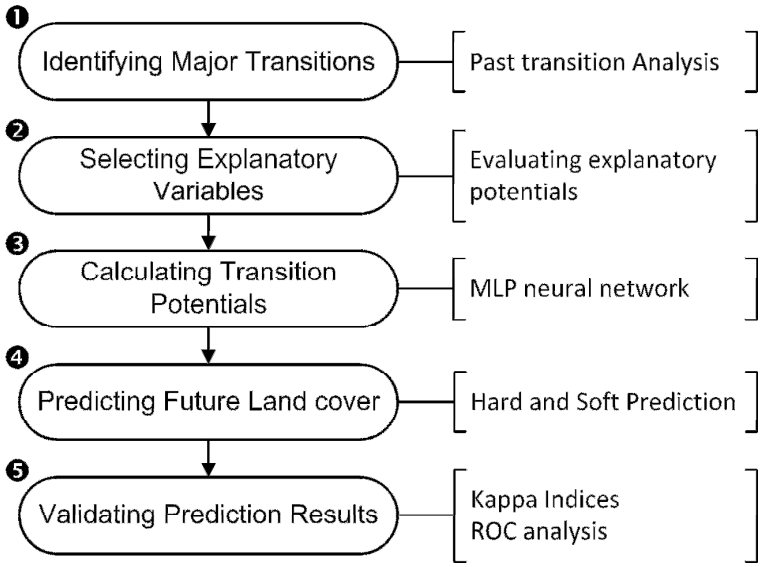

Figure 5.4 LCM simulation procedures

Table 5.1 Selected major land transitions from 1985 to 1995

Transition	Area of Change (km^2)	% Coverage
Forest to Paddy	182.72	1.8
Forest to Agriculture	120.96	1.2
Forest to Urban	37.28	0.4
Paddy to Urban	134.88	1.4
Barren to Urban	18.56	0.2

There are many factors affecting the land cover transition such as climate change, policy, economics, culture, and anthropogenic activities. All these factors have their weight of influence in the land cover transition (Schneider and Pontius, 2001; Pijanowski et al., 2002). Thus the explanatory variables, which are the drivers of transitions, should be carefully selected. The Cramer's V coefficient is a useful criterion to evaluate a degree of association between the explanatory variables and land cover classes (Eastman, 2009; Oñate-Valdivieso and Sendra, 2010). The Cramer's V coefficient varies from 0 (no association) to 1 (perfect association). Generally in land change modeling the coefficient of 0.15 or higher means that the variable is useful, and the coefficenter of 0.4 or higher means that the variable is good to use in the LCM (Eastman, 2009). All possible candidates of the explanatory variables were tested and selected by the above criterion of the Cramer's V coefficient (Table 5.2).

Table 5.2 Degree of association (Cramer's V) between the selected explanatory variables and land use classes

Variable	Forest	Paddy	Agriculture	Urban	Barren
Topography	0.44	0.36	0.08	0.10	0.05
Slope	0.06	0.60	0.50	0.11	0.19
Distance from Urban	0.05	0.26	0.16	0.05	0.39
Distance from Road	0.04	0.29	0.24	0.02	0.11
Distance from Stream	0.02	0.22	0.19	0.04	0.08
All land cover to Urban	0.07	0.68	0.57	0.42	0.30

The LCM provides a MLP neural network tool to calculate each transition potential. The MLP neural network is widely used technique to solve non-linear complex problems such as hydrologic forcast, landscape classification, and land change prediction (Alvici and Franchini, 2011; Coleman, 2008; Pijanowski et al., 2002). The MLP neural network in the LCM has only one hidden layer and the selected explanatory variables from Step 3 are used as input layer. The output of MLP neural network is the potential map of each transition. The MLP neural network was trained with three stopping criteria: the root mean square (RMS) of 0.01, the iterations of 10000, and the accuracy rate of 100%. The maximum sample size was 117 and the MLP training stopped after 10000 iterations with the accuracy rate of 89.5%. The calculated transition potentials are shown in Figure 5.5.

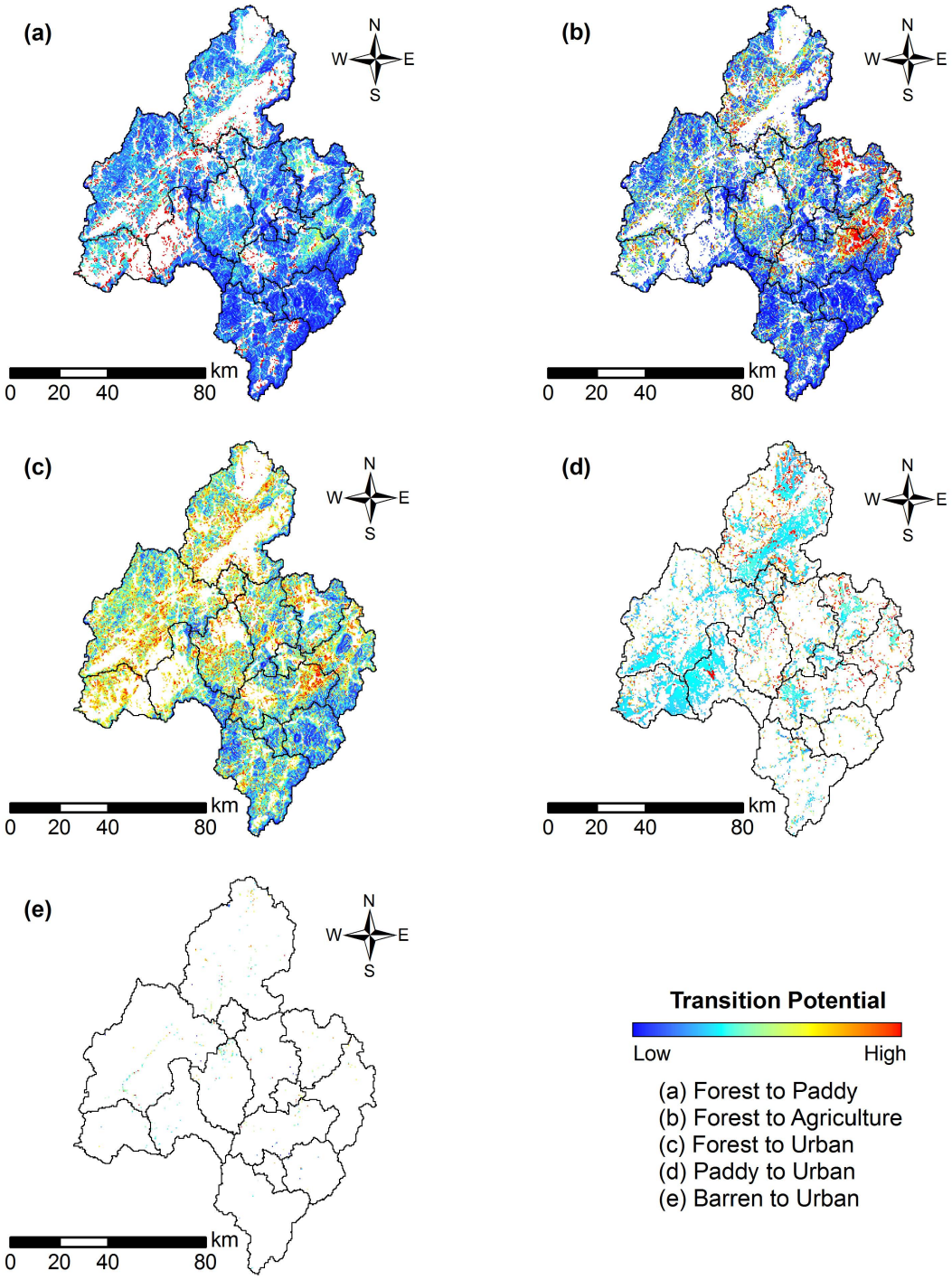

Figure 5.5 Calculated transition potential maps: (a) Forest to Paddy, (b) Forest to Agriculture, (c) Forest to Urban, (d) Paddy to Urban, (e) Barren to Urban.

The LCM supports 'hard prediction' having the same land cover classes with the input and 'soft prediction' showing a continuous distribution of vulnerability of the land cover change (Eastman, 2009). The LCM uses Markov Chain analysis for the hard prediction. The predicted land cover map of the year of 2000 was validated with the actual land cover map produced in 2000 (Figure 5.6a and 5.6b). In order to validate the predicted land cover, the standard Kappa index (Kstandard) and two alternative Kappa indices (Kno, and Klocation) suggested by Pontius (2000) were used. Kstandard is the degree of agreement varying from 0 (poor agreement) to 1 (perfect agreement). Kno and Klocation also show a value ranging from 0 to 1. Kno represents the overall accuracy of a simulation run and Klocation indicates the ability of the model to predict location of changed cell (Pontius, 2000). The overall percentage of correct cells is 89.4%, the calculated Kstandard is 0.83, and Kno and Klocation are equally 0.88, respectively. All the numbers show a strong agreement between the actual land cover and the predicted land cover. The validation of each transition type was performed by the ROC analysis (Pontius and Schneider, 2001) using the result of the soft prediction. The soft prediction map is created by aggregation of the modeled transition potentials by MLP (Figure 5.6c). Pontius and Schneider (2001) introduced the ROC technique for validation of land cover transition and its suitability. A perfect prediction has a ROC value of 1 and a random location of input has a ROC of 0.5. The ROC value greater than 0.8 indicates a strong agreement of prediction (Eastman, 2009).

The results of ROC analysis are summarized in Table 5.3. In case of the transitions from Barren and Paddy to Urban, the ROC values are relatively lower than the other transitions. It can be inferred that there are unknown explanatory variables affecting these transitions.

Table 5.3 ROC values for actual transition from 1995 to 2000

Transition	ROC
Forest to Paddy	0.874
Forest to Agriculture	0.875
Forest to Urban	0.855
Paddy to Urban	0.643
Barren to Urban	0.564

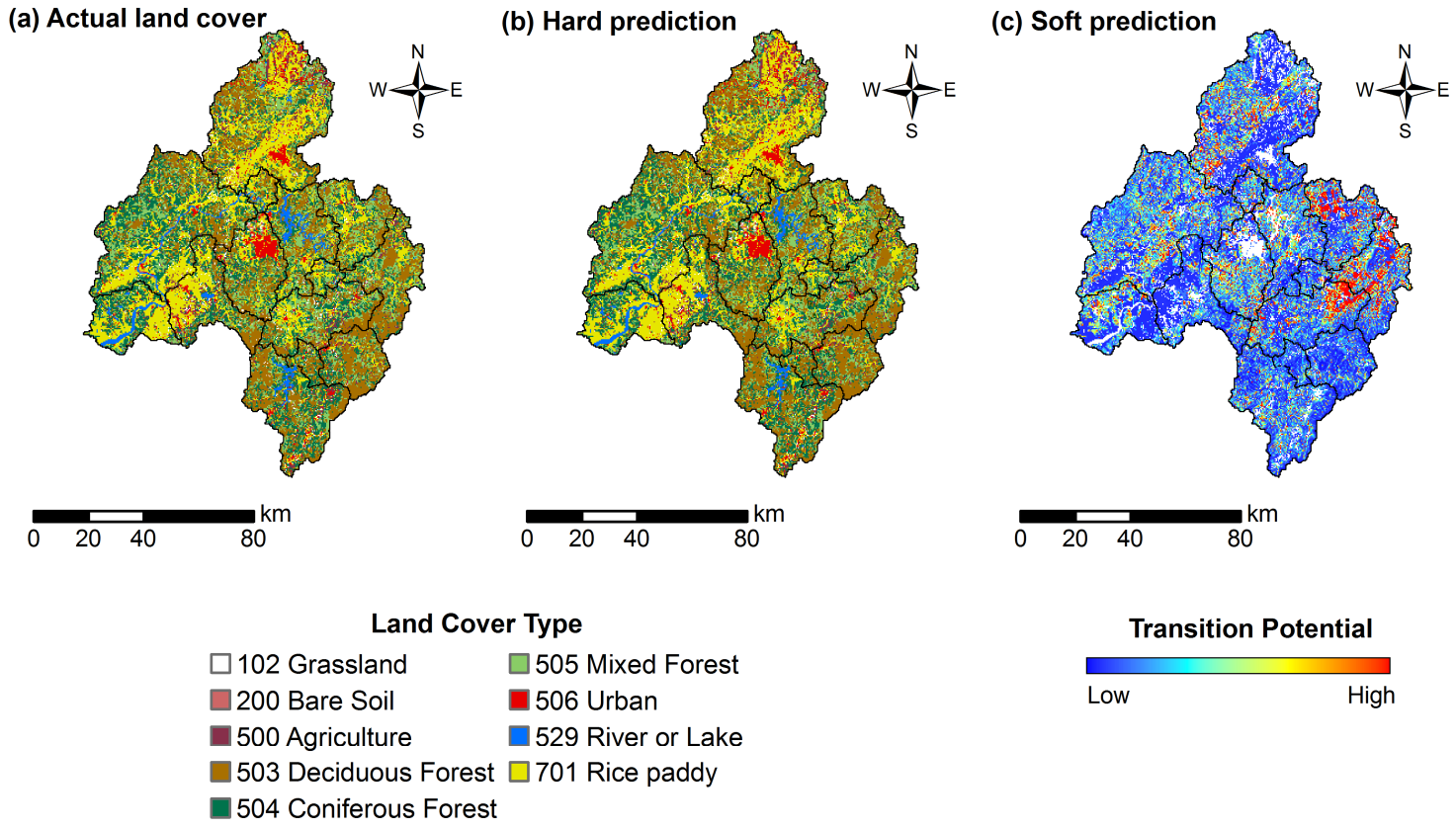

Figure 5.6 Comparison between actual and predicted land cover in 2000: (a) Actual land cover, (b) Hard prediction, (c) Soft prediction.

The validated land cover transition model was used to generate future land cover maps with a 5-year interval as an input of the VELAS model. Table 5.4 shows a summary of the land cover transition result. Land cover classes, which are not associated with the selected transition types, are not listed. The Forest and Barren land cover classes are decreased while the others are increased. The Forest is the most dorminant class of the land cover change for the Geum River Basin although the percentage of change is smaller than the other classes.

Table 5.4 Summary of the land cover transition from 2000 to 2050

Land Cover	Area in 2000 (km^2)	Area in 2050 (km^2)	Area of change (km^2, (%))		% coverage
Forest[a]	6,393	5,422	971	(-15.2%)	9.7
Paddy	2,241	2,823	582	(26.0%)	5.8
Urban	303	664	361	(119.1%)	3.6
Agriculture	557	594	37	(6.5%)	0.3
Barren	55	52	3	(-5.2%)	0.0

[a]Forest includes all forest types: coniferous, deciduous, and mixed.

5.4 Prediction Results from VELAS

The prediction procedures of VELAS followed the same steps explained in Chapter 2. Groundwater model was not included in the prediction model because the extent and geology of the Geum River Basin are too vast and complex to be conceptualized as a groundwater model. The VELAS model was

calibrated and validated during the year of 2001. A dummy year simulation was applied to decide the initial grid of SMD. The results for 2001 were validated with the same validation procedures in Chapter 4. The observed runoff of each sub-watershed was compared with the calculated runoff for the year 2001 (Figure 5.7). The overall correlation is high with R^2 value of 0.79 for all 14 watersheds in the year of 2001.

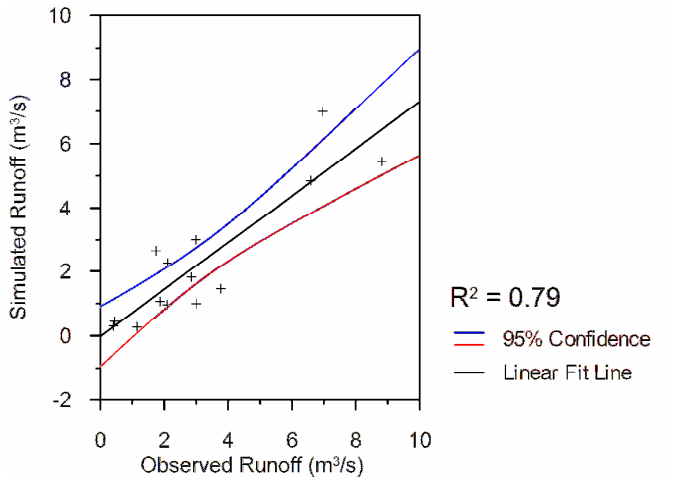

Figure 5.7 Correlation between the observed and simulated runoffs in 2001.

After the validation process, the model predicted the future hydrologic feedbacks from the year of 2011 to 2050. Figure 5.8 shows the annual variations of interception, runoff, evapotranspiration, and recharge. The overall trend of each component follows the precipitation change trend showing a decrease until the late 2020's and an increase from the early 2030's in Figure 5.3 because the precipitation is the strongest influencing factor in the hydrologic system.

However, the interception and runoff curves show a slightly different trend after the early 2030's due to the impact of land cover change. As shown in Figure 4.1, 4.2, and 4.3, the high amount of interception occurs on the forest area and the high amount of runoff occurs on the paddy or urban area. Therefore, a decrease of forest area induces a decrease of interception and an increase of paddy and urban areas leads an increase of runoff. A comparison between two interception and runoff events in the year of 2012 and 2047 in Figure 5.8a and 5.8b is an example of the land cover impact. The annual amounts of precipitation in 2012 and 2047 are similar: 941 mm and 944 mm, respectively. The amount of interception in the year of 2012 is 119 mm (❶) and the amount of runoff is 106 mm (❷). The area of forest decreases from 6,183 km^2 to 5,423 km^2; the area of paddy increases from 2,369 km^2 to 2,823 km^2; and the area of urban increase from 385 km^2 to 664 km^2 in the year of 2047. The amount of interception in the year of 2047 with the change of land cover is 81 mm, which is 32% decrease, (❸) and the amount of runoff is 182 mm, which is 72% increase, (❹).

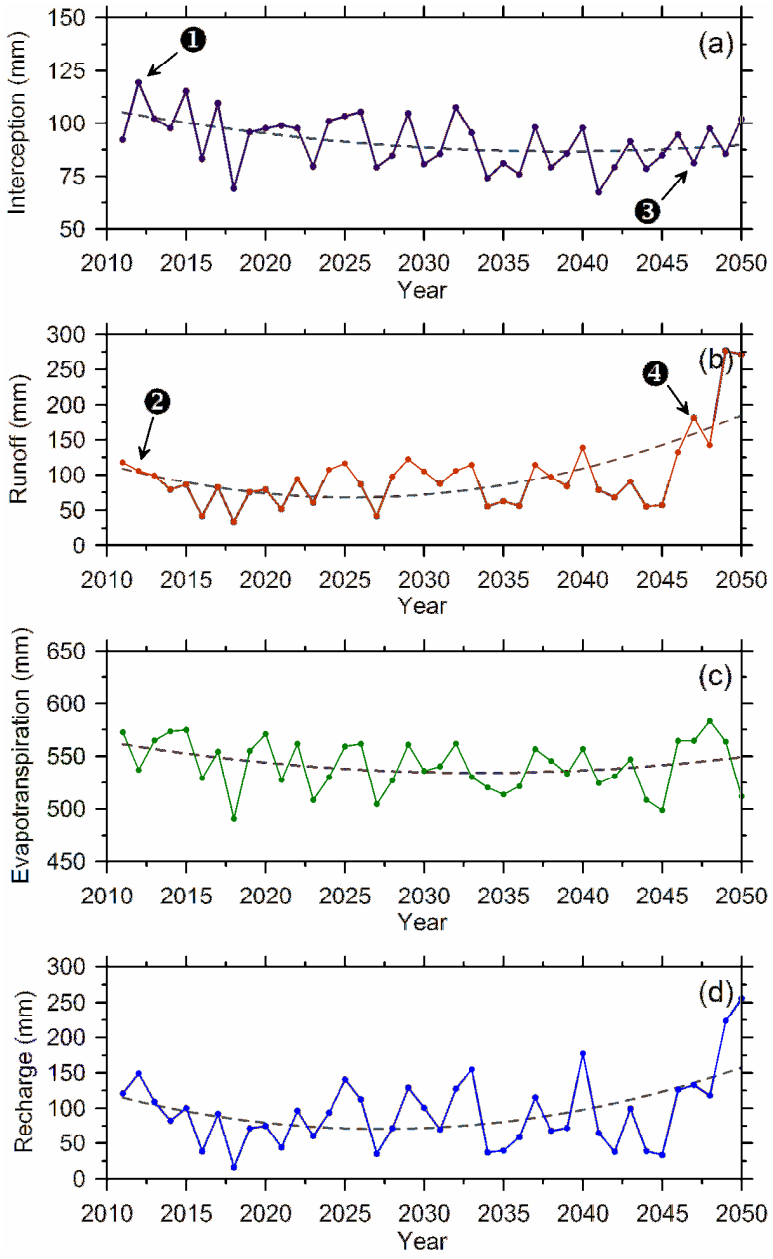

Figure 5.8 Predicted hydrologic feedbacks from the year of 2011 to 2050: (a) Interception, (b) Runoff, (c) Evapotranspiration, (d) Recharge.

CHAPTER 6

COMPARISON TO WetSpass MODEL

6.1 WetSpass Model

WetSpass (Water and Energy Transfer between Soil, Plants and Atmosphere under quasi Steady State) developed by Batelaan and De Smedt (2001), is a quasi steady state water balance model using long-term average weather data. The model is based on WetSpa, a time dependent spatial distributed water balance model (Batelaan et al., 1996) and implemented as a tool for ESRI ArcView 3.x environment using Avenue, an object-oriented programming language for ArcView (Batelaan and Woldeamlak, 2007). In this study, WetSpass was rewritten with Python version 2.5, a general purpose high level programming language, to make WetSpass supports the latest version of ArcGIS and operating system.

6.2 WetSpass Methodology

The WetSpass model estimates long-term average spatial patterns of surface runoff, actual evapotranspiration, interception, and groundwater recharge (Batelaan and De Smedt, 2007). The following description of the methodology is based on Batelaan (2006).

The WetSpass model employs a general water balance equation expressed as

$$P = RO + ET + RCH \quad (6.1)$$

where P is the precipitation, RO is the surface runoff, ET is the evapotranspiration, and RCH is the recharge. The WetSpass model classifies the surface environment into four different types of surface model: vegetated soils, bare soils, open water, and impervious surfaces. Thus, each surface model has an independent water balance equation:

$$P = I + RO_v + TR_v + RCH_v \quad \text{for vegetated soils} \quad (6.2)$$

$$P = RO_s + EV_s + RCH_s \quad \text{for bare soils} \quad (6.3)$$

$$P = RO_o + EV_o + RCH_o \quad \text{for open water} \quad (6.4)$$

$$P = RO_i + EV_i + RCH_i \quad \text{for impervious surfaces} \quad (6.5)$$

where I is the interception, assigned to be constant by the vegetation type, TR is the transpiration, and EV is the evaporation (Batelaan and De Smedt, 2001). Each parameter is handled as a raster grid with its attribute table. Therefore, the total water balance in a raster cell during a hydrologic period can be expressed as:

$$ET_c = a_v ET_v + a_s EV_s + a_o EV_o + a_i EV_i \quad (6.6)$$

$$RO_c = a_v RO_v + a_s RO_s + a_o RO_o + a_i RO_i \quad (6.7)$$

$$RCH_c = a_v RCH_v + a_s RCH_s + a_i RCH_i \quad (6.8)$$

where a_v, a_s, a_o, and a_i are the areal fractions for each surface condition. The areal fractions are determined in the model simulation by a pre-defined table containing individual areal fractions for each land use type (Batelaan, 2006).

For vegetated soils given in Equation (6.2), the interception, I, is calculated with a constant percentage from the vegetation type specified in a land use map attribute table:

$$I = C_{ip} \times P \tag{6.9}$$

where I is the interception [mm], C_{ip} is the constant percentage of interception by a vegetation type [-], and P is the precipitation [mm]. The surface runoff, RO_v, is calculated in two stages. The potential surface runoff, $RO_{v\text{-}pot}$, is calculated in the first stage:

$$RO_{v\text{-}pot} = C_{Sv} \times (P - I) \tag{6.10}$$

where $RO_{v\text{-}pot}$ is the potential surface runoff [mm], C_{Sv} is the runoff coefficient that is a function of vegetation type, soil texture, and slope [-]. Since this potential surface runoff simulates only on groundwater saturated areas, in the second stage the actual surface runoff, S_v, is calculated for recharge areas by taking into account differences in precipitation intensities in relation to soil infiltration capacities:

$$RO_v = C_{Hor} \cdot RO_{v\text{-}pot} \tag{6.11}$$

where C_{Hor} is the coefficient that parameterizes the part of the seasonal precipitation which contributes to the Hortonian surface runoff. It can be derived by estimating the fraction of seasonal precipitation with intensity higher than the infiltration capacity of a particular soil type. The potential transpiration, PTR_v, is determined as

$$PTR_v = c \cdot EV_o \tag{6.12}$$

where c is the vegetation coefficient [-] and EV_o is the Penman open water evaporation [mm]. The vegetation coefficient can be derived from the Penman-Monteith equation and the Penman open water evaporation equation:

$$c = \frac{1 + \frac{\gamma}{\Delta}}{1 + \frac{\gamma}{\Delta}(1 + r_c / r_a)} \tag{6.13}$$

where Δ is the slope of the saturation vapor pressure curve [kPa/°C], γ is the psychrometric constant [kPa/°C], r_a is the aerodynamic resistance [s/m], and r_c is the canopy resistance [s/m]. The actual transpiration, ATR_v, is equal to the potential transpiration in vegetated groundwater discharge areas because soil water availability is unlimited. However, the actual transpiration for vegetated areas where the groundwater level is below the root zone is calculated as

$$ATR_v = f(\theta) \cdot PTR_v \tag{6.14}$$

where $f(\theta)$ is a function of the water content (θ) in root zone and determined as

$$f(\theta) = 1 - a^{\frac{w}{PTR_v}} \tag{6.15}$$

where a is a calibrated parameter related to the soli texture, which increases with decreasing saturated hydraulic conductivity, and w is the available water for transpiration [mm]

$$w = P_n + n(FC - WP) \cdot Z_r \cdot 1000 \tag{6.16}$$

where P_n is the precipitation of hydrological season consisting of n months [mm], FC is the field capacity, WP is the permanent wilting point, and Z_r is the rooting depth [m].

The water balance of bare soil surfaces is represented in Equation (6.3). The surface runoff, RO_s, is calculated with the same manner of the vegetated surface runoff (Equation (6.2) and (6.3)). The evaporation, EV_s, is also calculated as

$$EV_s = f(\theta)\, EV_o \tag{6.17}$$

where $f(\theta)$ is a function, which is defined in Equation (6.15) and in which PTR_v is replaced by EV_o.

The calculation of the water balance for open water surfaces, Equation (6.4), is relatively simple. The surface runoff, RO_o, is calculated as

$$RO_o = P - EV_o \tag{6.18}$$

The water balance of impervious surfaces is represented in Equation (6.5). The surface runoff can be calculated as a coefficient times the precipitation:

$$RO_i = C_{Si} \cdot P \tag{6.19}$$

where C_{Si} is a runoff coefficient for impervious surfaces [-], which is a function of the type of impervious cover and slope. The evaporation, EV_i, is calculated as

$$EV_i = C_{Ei} \cdot (P - RO_i) \tag{6.20}$$

where C_{Ei} is an empirically determined constant.

Open water evaporation, EV_o, is one of input datasets of the WetSpass model but WetSpass does not calculate it during the simulation processes. Thus, it should be calculated and created as an input before the WetSpass modeling. For the calculation of open water evaporation with the limited

datasets, Valiantzas's simplified Penman evaporation equation (Valiantzas, 2006) was used:

$$EV_o \approx 0.051(1-\alpha)R_S\sqrt{T+9.5} - 0.188(T+13)\left(\frac{R_S}{R_A} - 0.194\right) \times\left(1 - 0.00014(0.7T_{\max} + 0.3T_{\min} + 46)^2\sqrt{\frac{RH}{100}}\right) + 0.049(T_{\max} + 16.3)\left(1 - \frac{RH}{100}\right)(a_u + 0.536u_2) + 0.00012 \cdot Z_h \quad (6.21)$$

where α is the albedo [-], T_{max} is the mean maximum air temperature [°C], T_{min} is the mean minimum air temperature [°C], T is the mean air temperature [°C], R_S is the solar radiation [MJ/m^2/day], R_A is the extraterrestrial radiation [MJ/m^2/day], RH is the relative humidity [%], a_u is a constant, 0.0 in Linacre (1993), 0.5 in Penman (1956) and Cohen et al. (2002), and 1.0 in Penman (1948), for the original Penman wind function, u_2 is the wind speed at 2 m height [m/s], and Z_h is the elevation of the weather station [m].

An equational comparison between VELAS and WetSpass is summarized in Table 6.1.

Table 6.1 Equational comparison between VELAS and WetSpass

Model	Interception	Runoff	Evapotranspiration	Recharge
VELAS	Aston and von Hoyningen-Huene method[a]	NRCS-CN method[b]	FAO-PM[c]	RCH = |-SMD| when $\delta_{gw} < 1$ or RCH = Equation (2.37) when $\delta_{gw} \geq 1$
WetSpass	Constant fraction of precipitation[d]	Modified rational method[e]	PM[f]	Residual term RCH = P – I – RO - ET

[a]Aston (1979) and von Hoyningen-Huene (1983).
[b]USDA-NRCS (2004).
[c]Allen et al. (1998).
[d]Batelaan (2006).
[e]Batelaan (2006).
[f]Monteith (1965).

6.3 WetSpass Algorithm and Implementation

The WetSpass algorithm is relatively simpler than the VELAS algorithm. Figure 6.1 illustrates an algorithm for the WetSpass model. WetSpass simulates the water balance of winter as a dry season first, and summer as a following wet season (Batelaan and Woldeamlak, 2007). For each season, the interception is calculated as a first step, then the runoff, transpiration and evaporation calculations are followed. In WetSpass, the evapotranspiration is the sum of interception, transpiration, and evaporation. The groundwater recharge for each surface model is calculated as the residual term. If the groundwater model exists, WetSpass and MODFLOW interoperate until the head change difference reaches the convergenece criterion by user. Two sets of raster datasets for winter and summer are required to perform a year simulation. The simulation results are saved as two different format, ESRI's raster grid and ASCII, to support flexible visualization.

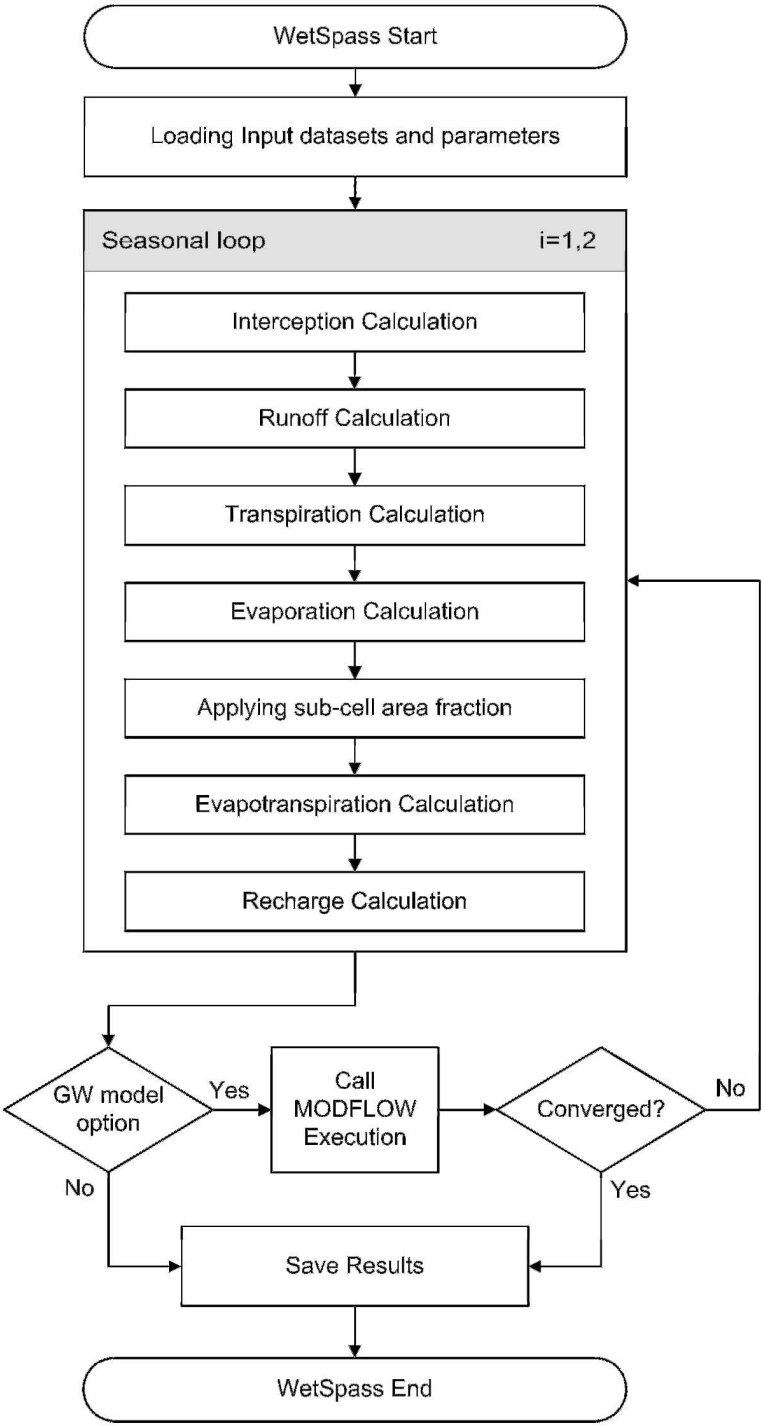

Figure 6.1 WetSpass algorithm flowchart.

The WetSpass model was implemented as a customized geoprocessing toolbox in ArcGIS 9.x using Python (Figure 6.2). The WetSpass tool pack includes two python script files: WetSpass-MODFLOW and WetSpass-Timeseries Batch. The WetSpass-MODFLOW script is for the one year simulation with MODFLOW model and the WetSpass-Timeseries Batch script is for the annual base timeseries simulation without MODFLOW model.

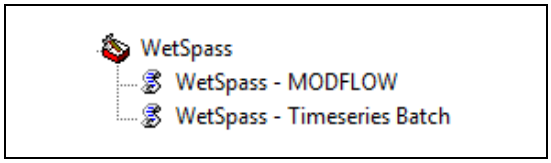

Figure 6.2 Installed WetSpass tool pack in ArcGIS 9.x.

The interface of the WetSpass script was coded with the arcgisscrpiting module of ArcGIS 9.x. The arcgisscripting module is a supporting module for Python programming to provide access point for all geoprocessing tools in ArcGIS (ESRI, 2008). Figure 6.3 is the user interface window of the WetSpass-MODFLOW tool. Every input dataset is categorized by its property and season with the category number (0 to 4). The category 0 is for the interoperation with the MODFLOW model and only be activated when the check box of the Run with MODFLOW menu is checked. The WetSpass requires land use, precipitation, potential evaporation, temperature, windspeed, and groundwater depth data for each season. The category 1 and 2 are for the

winter and summer datasets. Two season-independent raster datatsets, slope and soil, are handled with the category 3. The parameter tables to provide additional information of land use, runoff, and soil rasters are the catergory 4. The WetSpass-Timeseries Batch script has the same user interface except the option for the MODFLOW interoperation.

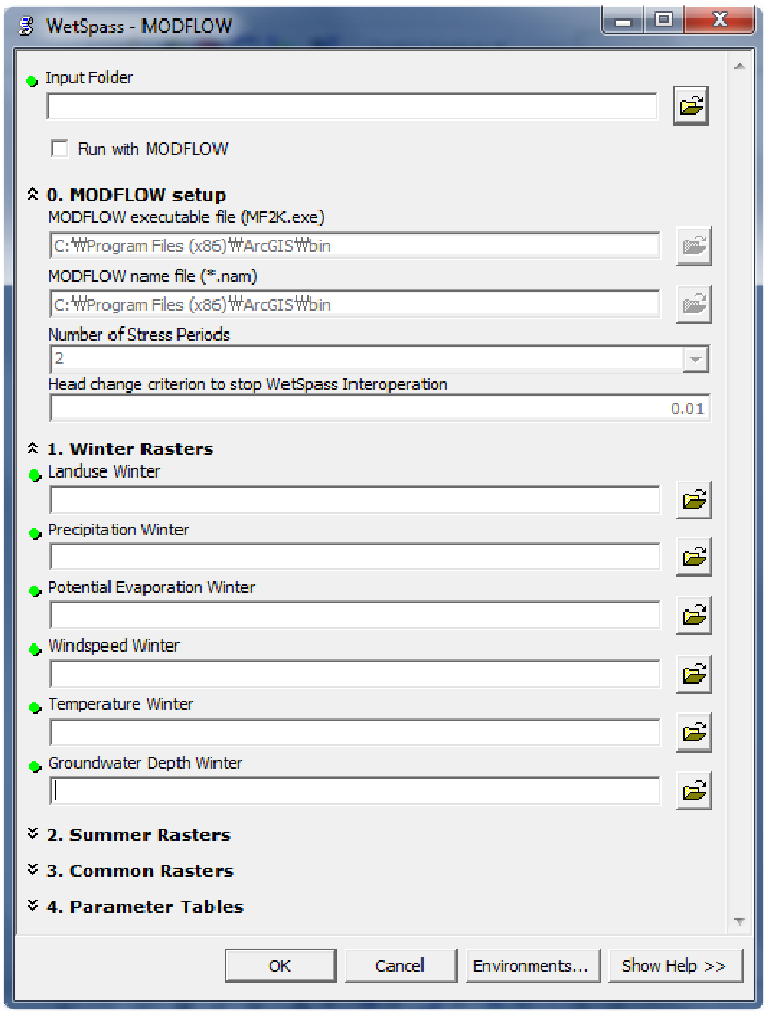

Figure 6.3 Interface window of the WetSpass tool.

6.4 Example Application of WetSpass to Jeju Island

For the purpose of basic understanding and testing of the model performances, WetSpass was applied to Jeju Island in Korea. This application was performed with the original WetSpass extension for ArcView 3.x. Jeju Island is a volcanic island located in the southernmost region of Korea. The Island mainly consists of highly permeable volcanic structures such as lava tubes, channels or clinkers. Since water from precipitation barely resides on the surface and mostly infiltrates into the aquifers or discharges directly to the ocean, the Island does not have any perennial streams or rivers, which makes groundwater to be the only source of water in the Island. Accurate estimation of groundwater recharge is therefore very crucial to develop a sustainable water management plan for people living in the Island.

6.4.1 Site Decription and Data

Jeju Island is located in the southernmost province of Korea, where its latitude is from 33°11'40"N to 33°33'54"N and the longitude is from 126°09'36"E to 126°56'45"E (Figure 6.4). The shape of the Island is elliptical with a long axis of 73 km and a short axis of 31 km and covers an area of 1,832 km^2. The island has sixteen watersheds based on topography and stream distribution and there are total nineteen weather stations including fifteen automated weather stations, which are operated by the Korea Meteorological Administration (KMA). Jeju Island has a mild oceanic climate condition with a distinct rainy

summer season from June to September. The annual average temperature is 15°C and its local variation is not more than 1°C. The humidity of Jeju Island shows 70% to 80% throughout the year and any monthly variation is barely observed. The average annual precipitation is 1,000 mm in the western region to 1,975 mm in the southern region and mostly concentrated in the rainy summer season from June to September. Jeju Island is a shield volcano covered by Quaternary basaltic lava (Hwang et al., 1994). Basalt covers over 90% of the island and is highly permeable. Multiple volcanic activities and lava flows formed a layered set of basalt having complex and highly permeable structures. Although a significant amount of rainfall occurs in the range of 1,060 mm/year to 3,593 mm/year (KOWACO, 2003), due to the high permeability of basaltic layers, there are almost no perennial streams in the island.

The data models for WetSpass variables were built on the two-dimensional rectangular space having a total number of 45,891 raster cells with 200 m × 200 m cell size. Since the WetSpass model conducts cell-by-cell calculations, every raster map has the same resolution and same number of cells. The land use map and soil cover raster map were obtained from the WAMIS. The study period from October 2006 to September 2007 was divided into two seasons, the winter season which is lasted from October 2006 to March 2007 and the summer season which is lasted from April 2007 to September 2007. Hydraulic datasets including weather and groundwater level data were also prepared for each season, respectively.

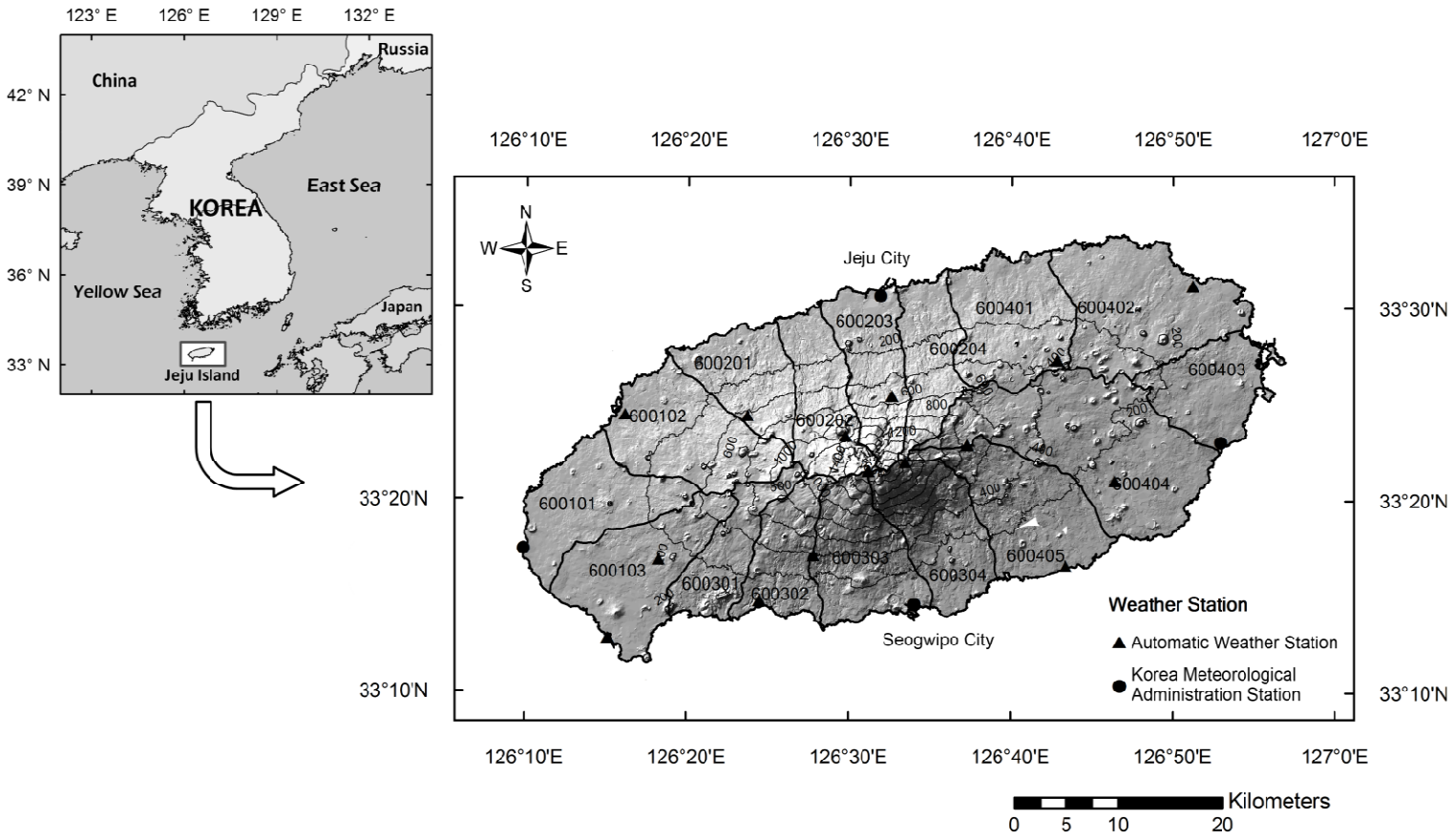

Figure 6.4 Shade relief map of Jeju Island delineated with watershed boundaries (Solid triangles and circles indicate locations of weather station).

Since WetSpass was developed in Belgium, it does not account for paddy field and larch forest. Therefore, the model was modified. The parameterized properties of rice paddy fields were added to the attribute table of the land use map containing vegetation parameters such as plant height and LAI (Kim et al., 2005). The added height of plant for rice paddy is 1 m and the LAI is 5. The land cover of the Island is dominated by agriculture (39%), forest (25%), and shrub (26%). The residential area (8%) is very limited along the coast where water is provided from spring waters (Figure 6.5). The major soil type that covers an area of 1,254 km^2 is silty clay loam (68%) (Figure 6.6). The cetral area is mainly covered with clay loam (19%) and sandy loam (2%). Clay soils (10%) resulting in higher runoff and lower recharge are dominant over the western region of the Island. Sand (1%) is rarely observed in limited areas.

The precipitation, temperature, relative humidity, and wind speed data were obtained from nineteen weather stations that are evenly distributed and operated either automatically or manually by KMA (Figure 6.4). Since KMA measures wind speed at 10 m above the ground surface, the wind speed data were adjusted with Equation (6.22) (Allen et al., 1998) to follow the requirement of WetSpass, in which the measurement height is 2 m.

$$u_2 = u_z \frac{4.87}{\ln(67.8 \cdot z - 5.42)} \tag{6.22}$$

where u_2 is the wind speed at 2 m above ground surface [m/s], u_z is the measured wind speed at height z m above ground surface [m/s].

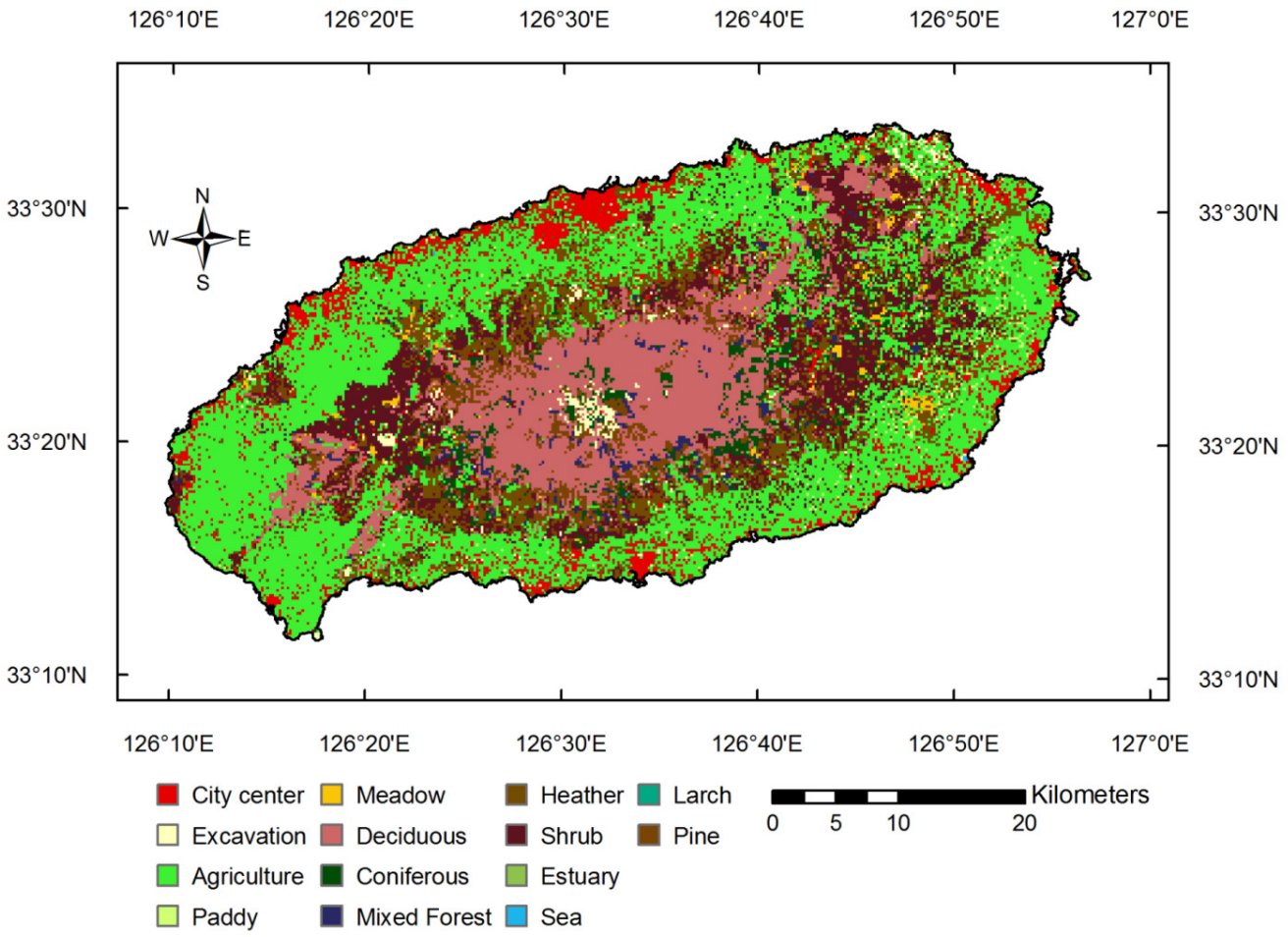

Figure 6.5 Land cover map of Jeju Island.

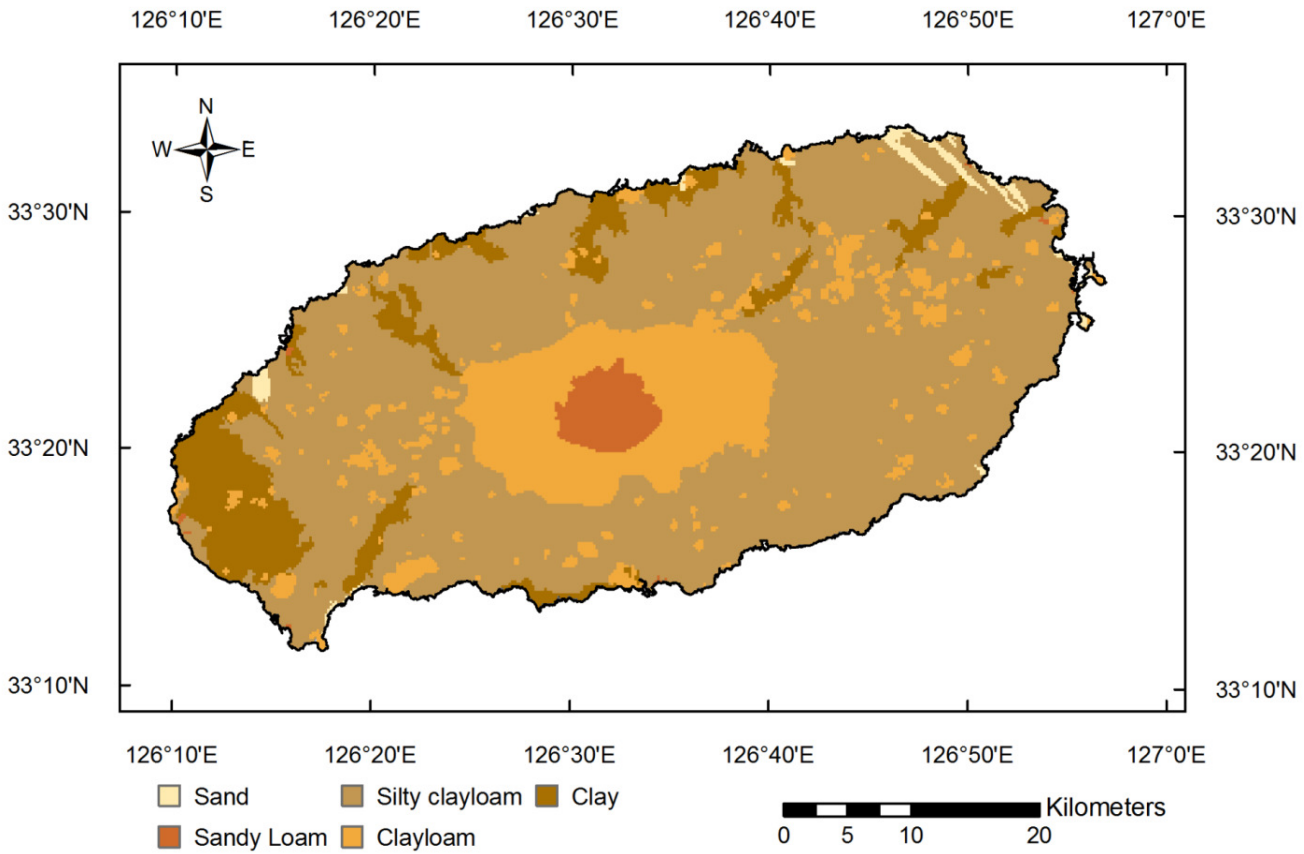

Figure 6.6 Soil map of Jeju Island.

6.4.2 Application Results

Figure 6.7 shows the distributed maps of water balance components for Jeju Island. The estimated annual mean runoff ranged from 7 mm/year to 3,057 mm/year (Figure 6.7a). The highest rate of runoff was observed in the urban areas where the impervious surface of Jeju City and Seogwipo City is dominant. The western region of the Island and the mid-mountainous area, of its elevation ranges from 600 m to 1,200 m, showed a high rate of runoff due to the clay soil and the steep slopes. Despite the steep slopes, average 24% up to 80% for 100 m elevation interval, the higher altitude area over 1,200 m showed a relatively lower amount of runoff. This is because the sandy loam covering the top of the mountain has a low runoff coefficient. The total evapotranspiration is higher at the northern and southern regions, where the agricultural land is dominant, with the highest rate of 1,413 mm/year while the one at the eastern and western regions has the lowest rate of 374 mm/year (Figure 6.7b). For the annual mean precipitation of 2,593 mm/year during the study period, the estimated average annual recharge was 1,055 mm/year ranging from -533 mm/year to 4,172 mm/year (Figure 6.7c). The eastern and northern regions of the Island showed relatively higher rates of recharge in comparison to the western region. The model results also showed that 69% of the total recharge occurred in the elevation over 200 m, which would be a critical groundwater recharge zone.

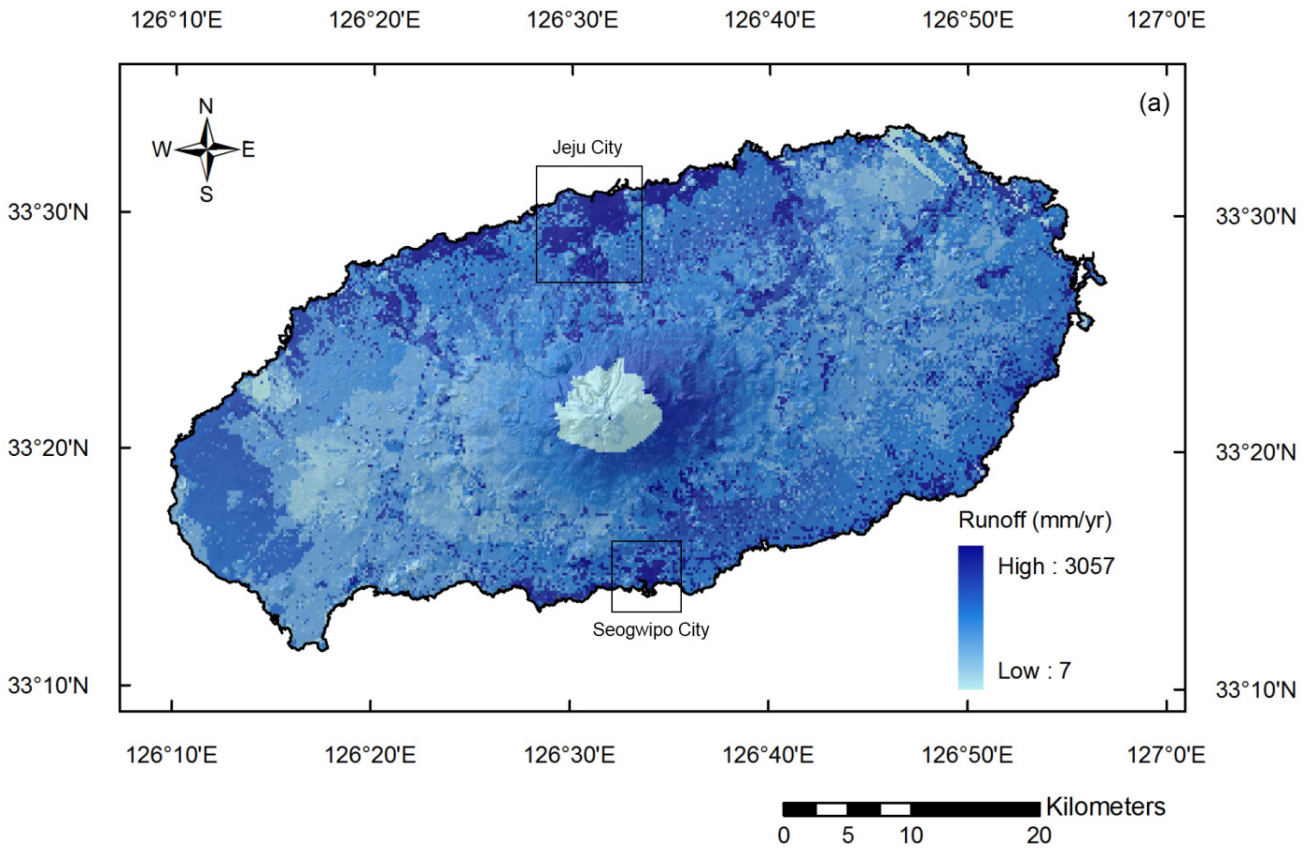

Figure 6.7 Distributed maps of simulated water balance components for Jeju Island: (a) Runoff, (b) Evapotranspiration, (c) Recharge.

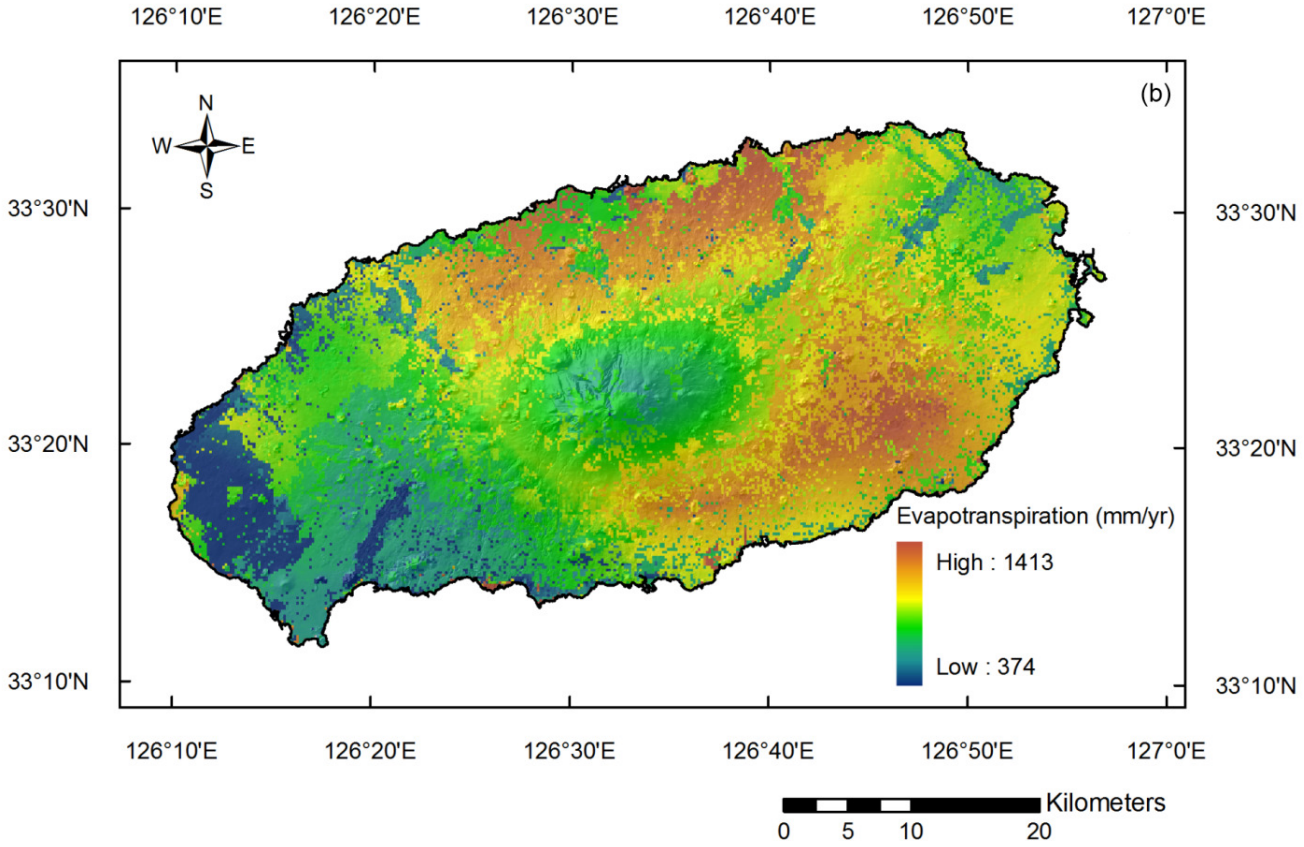

Figure 6.7 Continued.

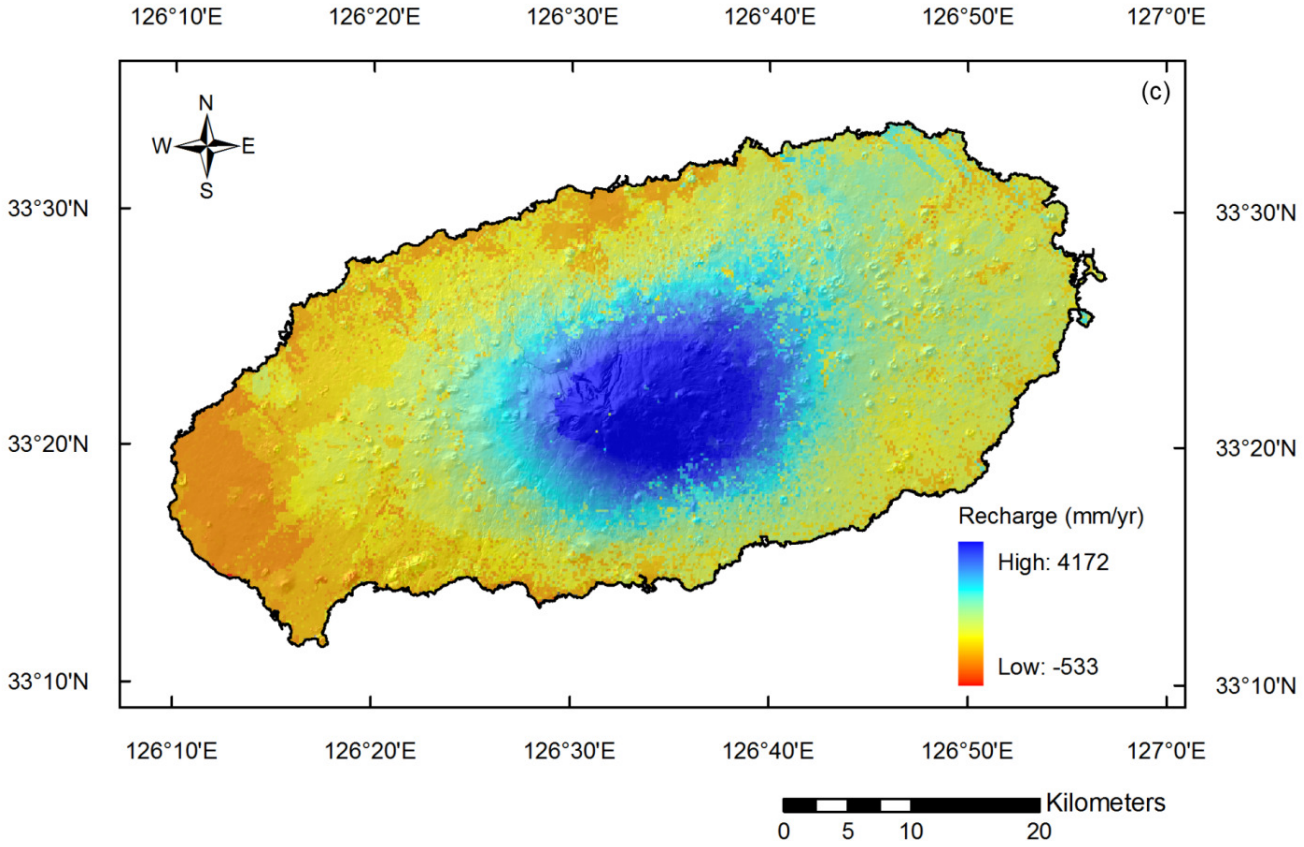

Figure 6.7 Continued.

Figure 6.8 shows the annual mean water balance by the soil texture. The highest recharge occurred on the sandy loams in the central area while the clay soils received the lowest recharge in the western region of the Island. It is clear that the high precipitation rate and the dominant coverage of sandy loam in the central area produce the highest rate of recharge in the Island, while the dominance of clay soil cover in the western region of the Island results in the lowest recharge with higher rate of runoff. The annual mean water balance by the land cover is presented in Figure 6.9. The higher annual mean recharge was observed from the forest and bare areas (71% of the total recharge) throughout the Island. The urban area along the coast showed the highest runoff rate of 1,170 mm/year, 45% of the annual mean precipitation.

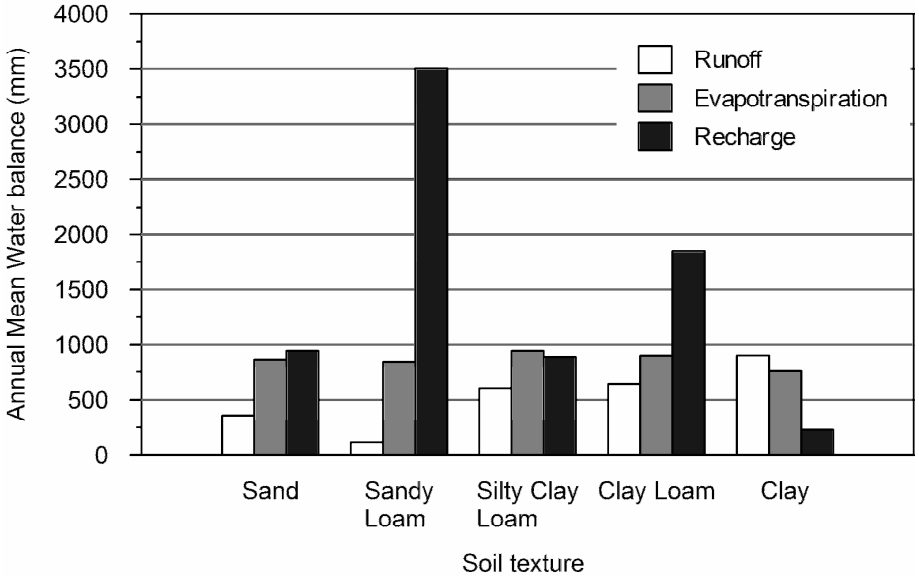

Figure 6.8 Annual mean water balance by soil texture.

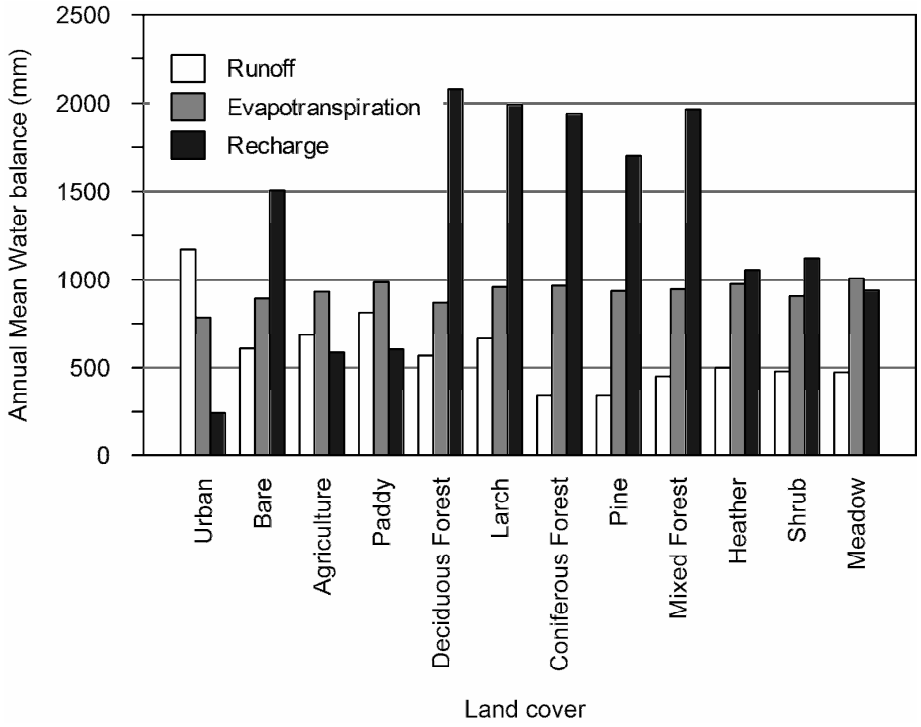

Figure 6.9 Annual mean water balance by land cover.

6.5 Results of WetSpass Application to the Geum River Basin

The WetSpass model was applied for the period from the year 2001 to 2010. Figure 6.10 shows the decadal average of the distributed hydrologic feedbacks for the Geum River Basin. The mid and southern regions of the basin show the higher amount of precipitation (Figure 6.10a). The higher amounts of interception occur on the forest area due to the vegetation property (Figure 6.10b). The range of interception amount is from 0 mm (urban) to 589 mm (forest). The estimated annual average runoff ranged from 6 mm to 1,084 mm

(Figure 6.10c). The highest rate of runoff was observed in the urban areas where the impervious surface of urban areas is dominant. The southwestern region of the basin where the most downstream area of the Geum River shows a high rate of runoff due to the low permeability of the soil. In WetSpass, the evapotranspiration is the sum of interception, transpiration and evaporation but Figure 6.10d only shows the sum of transpiration and evaporation in order to compare with VELAS. The evapotranspiration is higher at the open water area with the highest rate of 1,311 mm per year while the urban and bare areas have the lowest rate of 332 mm per year. The estimated average annual recharge was 217 mm ranging from 0 to 539 mm (Figure 6.10e). The distribution of recharge generally follows the distribution of precipitation and the higher rates of recharge occur on the forest area and high permeable soil area.

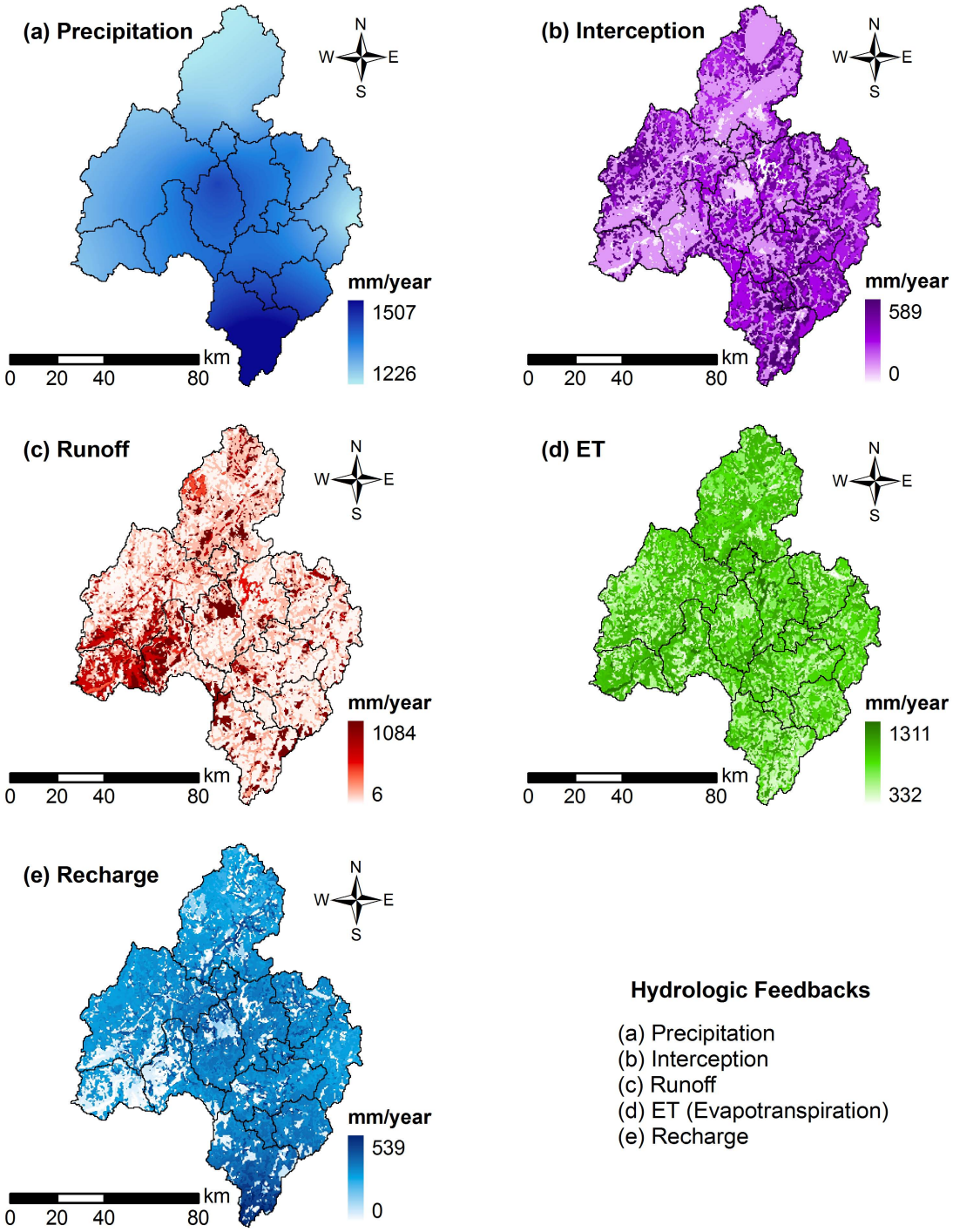

Figure 6.10 Decadal average (2001~2010) of the WetSpass simulation results: (a) Precipitation, (b) Interception, (c) Runoff, (d) Evapotranspiration, (e) Recharge.

6.6 Comparative Analysis

For the period from the year 2001 to 2010, the results of the VELAS model were compared with the WetSpass results. Table 6.2 shows a 10-year average fraction of interception, runoff, evapotranspiration, and recharge to the average precipitation. The factions of interception and runoff are significantly different. It is because the interception in WetSpass is calculated as a constant percentage from the precipitation (Batelaan, 2006) while VELAS considers that interception is a function of LAI and rainfall (Figure 6.11b). Thus, the amount of interception with the high amount of precipitation from WetSpass can be overestimated and the excessive amount of interception ultimately results in the underestimated amount of runoff. The runoff calculation in WetSpass considers land cover types, soil textures, slope, and vegetation types. It is the same as VELAS. However, WetSpass has only three types of vegetation (crop, grass, and forest) for the runoff calculation even though there are various species of vegetation having different runoff characteristics. For example, soybean and rice are crops but grows in different hydrologic circumstances affecting a different degree of impact to runoff. Unlike WetSpass, VELAS takes into account the impact of each indivisual vegetation type to runoff. The runoff result of VELAS shows relatively higher amount of runoff on the rice paddy area than the other agriculture area (Figure 6.11c) while WetSpass result shows similar amount of runoff for the rice paddy and agriculture area (Figure 6.11c). The evapotranspiration by WetSpass shows

more distinct amount contrast by the vegetation types than the VELAS evapotranspiration (Figure 6.10d and 6.11d). The recharge distributions of WetSpass and VELAS simulation show an inverse trend to the runoff distributions (Figure 6.10e and 6.11e). In case of the amount of recharge on the urban area, however, the VELAS result shows higher amount of recharge than the WetSpass result. It is because VELAS calculates evapotranspiration on the urban area under consideration of sub-cell vegetation and soil moisture variation while WetSpass regards evapotranspiration on the urban area as a constant fraction of residual water after runoff process.

Table 6.2 Comparison between VELAS and WetSpass

Model	Interception	Runoff	Evapotranspiration	Recharge
VELAS	10.3%	17.9%	48.9%	22.9%
WetSpass	18.8%	9.6%	55.0%	16.6%

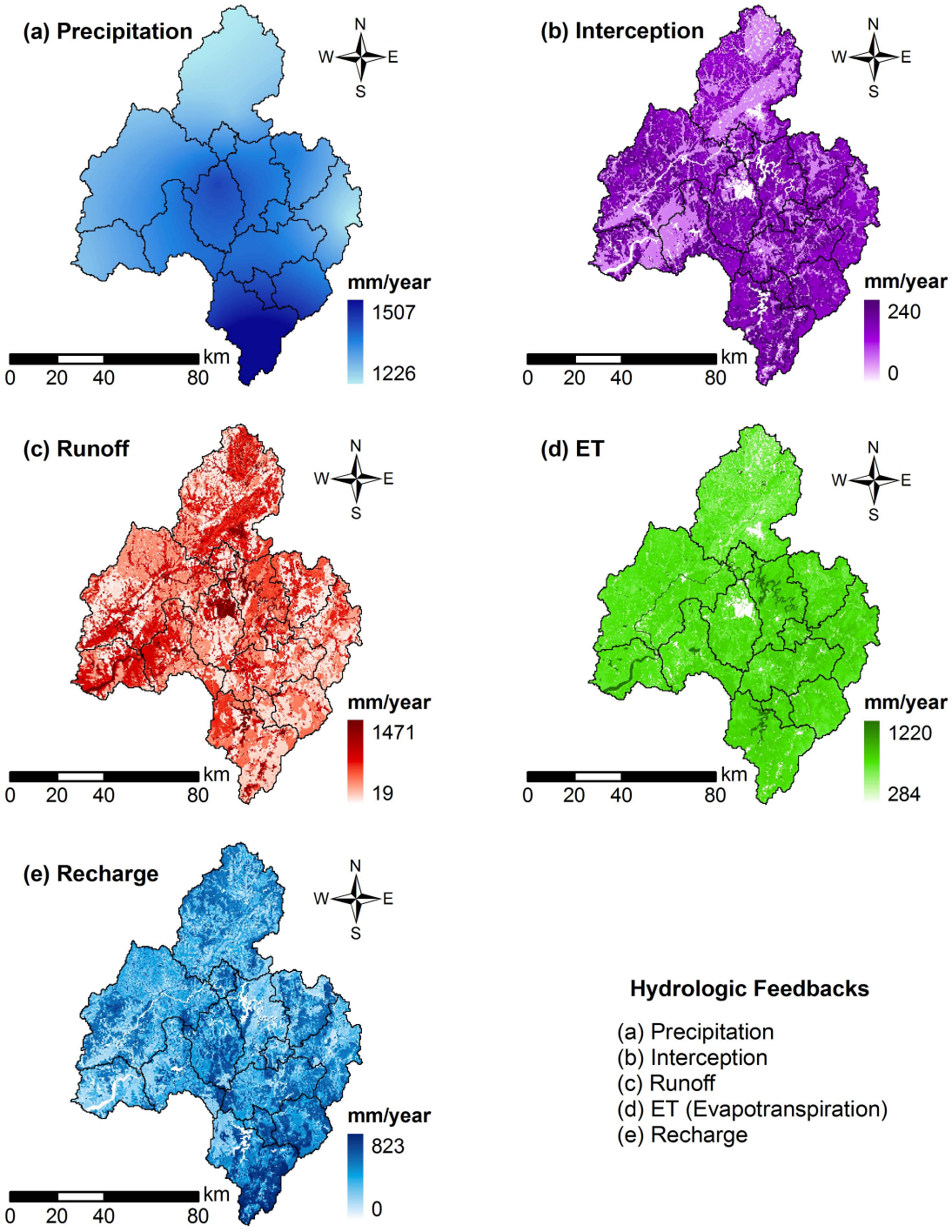

Figure 6.11 Decadal average (2001-2010) of the VELAS simulation results: (a) Precipitation, (b) Interception, (c) Runoff, (d) Evapotranspiration, (e) Recharge.

CHAPTER 6

CONCLUSIONS

The vegetation-land cover dynamics and the soil water balance are fully integrated in the VELAS model. The VELAS model is developed for advanced performance of spatial and temporal simulation of hydrologic feedbacks to the vegetation growth, land transition, soil water variation, and climate change in a watershed scale. The types and growth patterns of vegetation affect interception, runoff, evapotranspiration, and land transition. Rushton's soil water balance model is seamlessly integrated in the model and well communicated with the runoff and evapotranspiration calculations as a key element of the VELAS model. The results of the model application to the study site in the Geum River Basin show that VELAS has a capability to simulate complex feedbacks between hydrologic components. The successful application of sub-cell concept for a land cover transition in a single cell means that VELAS can be applied to any small area such as a patch of irrigation field although it is not in a watershed scale. The VELAS model also show a capability of providing information of the coupled surface and subsurface water interactions through the integration with MODFLOW. The simulated groundwater elevations from the calculated recharge in the study area showed a good agreement with the observed groundwater elevations.

A framework for a prediction of future hydrologic feedbacks was presented through an extended application to the Geum River Basin with the climate and land cover change models from 2000 to 2050. The land cover change by the LCM showed an increase of urban area by more than 100% and a decrease of forest by 15% in the Geum River Basin. The VELAS predicted that the increase of urban area would result in 32% of decrease of interception and 72% of increase of runoff by 2047.

The VELAS has some limitations for further development in the future. A cell-to-cell water transfer by surface runoff and lateral flow in a soil zone is not yet fully considered in the VELAS model. This limitation may be more critical for some regions where flash flooding occurs frequently. The impact of snow melt is not included as well this time. Therefore, application of VELAS to cold region or mountain range may increase uncertainty of the model results.

APPENDIX A

SOIL WATER BALANCE ALGORITHM

For a computational approach, Rushton (2006) suggested an algorithm of the soil water balance:

1. $In_i = P_i - I_i - RO_i + SURFSTOR_{i-1}$

   where In_i is the infiltration, $SURFSTOR_{i-1}$ is the near surface soil storage from the previous day.

2. If $In_i > PETi$ then $SURFSTOR_i = FRACSTOR(In_i - PET_i)$

3. AET_i calculation

   a. If $SMD_{i-1} < RAW_i$ or $In_i > PET_i$

      then $AET_i = PET_i$

   b. If $TAW_i \geq SMD_{i-1} \geq RAW_i$ and $In_i < PET_i$

      then $AET_i = In_i + K_{ri}(PET_i - In_i)$

      where $K_{ri} = (TAW_i - SMD_{i-1})/(TAW_i - RAW_i)$

   c. If $SMD_{i-1} \geq TAW_i$ and $In_i < PET_i$

      then $AET_i = In_i$

4. $SMD_i = SMD_{i-1} - In_i + SURFSTOR_i + AET_i$

5. If $SMD_i < 0$

   then $PERC_i = ABS(SMD_i)$ and $SMD_i = 0$

   where $PERC_i$ is the amount of water from the bottom of soil zone.

APPENDIX B

PYTHON CODE FOR THE VELAS MODEL

```
#===============================================================================
# Main Interface of VELAS
#   Python 2.5 code by Changhui Park <Changhui.Park@umkc.edu>
#
#   Date 07.20.2011
#
# Hydro Lab.
# Department of Geosciences, University of Missouri – Kansas City
#===============================================================================

import VELAS
import os

if __name__ == '__main__':

   # Setting up the main work folder
   MainFpath = os.getcwd()

   # Creating VELAS instance
   WB_model = VELAS.VELAS(MainFpath)

   # VELAS Execution
   Start = 2001
   End = 2050
   for year in range (Start,End + 1):
      WB_model.iyrRun(year)
```

```
#===============================================================================
# The Vegetation – LAnd cover – Soil water dynamics model : VELAS
#    Python 2.5 code by Changhui Park <Changhui.Park@umkc.edu>
#    Date 07.20.2011
#
# Hydro Lab.
# Department of Geosciences, University of Missouri - Kansas City
#===============================================================================

from numpy import array,zeros,where,copy
from calendar import isleap
import csv,os
import SceneGen
import RainINT, NRCSCNrunoff, EVTLIB, ISWB, ActualRCH, MODFdriver
import IOman, IOF_init

class VELAS:
# Default Constants
   # NRCS CN method Ia/S value: 0.2 for original or 0.05 for better fit
   IaS_ratio = 0.05
   # NoData value for ESRI ASCII grid
   intNoData = -9999
   # Bare Soil EV coefficient when using PM-ETo(FAO56) 1.10 for temperate and 1.05
     for semi-arid
   Ke = 1.10
   # Adjustment coefficient (0.16 to 0.19) interior to coastal area. default=0.16
   kRs = 0.16
   # Recharge dalay time [day] - It should be decided at the 2nd pre-run phase.
   deltaGW = 1
   # Execution Mode
   initialRun = True
   # base array for creating of an empty array. It is extracted from the Ibound array.
   ActiveArrayShape = None
   # the number of active cells by ActiveArray
   numcells = None

# Input from user
   # Model boundary which is consisted of 0(inactive) and others(active)
   Ibound = None
   # Daily Precipitation [mm]
   PPT = None
   # Daily mean Air Pressure [kPa]
   AirPR = None
   #Daily Maximum temperature [C]
   Tmax = None
   #Daily Minimum temperature [C]
   Tmin = None
   #Daily Mean temperature [C]
   Tavg = None
```

```
    # Daily mean wind speed at 2 m above ground surface [m/s]
    u2 = None
    # Daily mean Relative Humidity [%]
    RHmean = None
    # Slope [-] 0.0 ~ 1.0 (0 for 0% and 1 for 100%)
    SLP = None
    # Soil Texture raster (1 to 12 fixed)
    SOL = None
    # Land Cover raster (by user)
    LCV = None
    # Vegetation raster (by user)
    VEG = None
    # Surface Elevation from Digital Elevation Model [m]
    DEM = None
    # radian latitude
    rLat = None
    # Groundwater level from MODFLOW simulation [m]
    GWL = None
    # GWL observation time-series data with cell-ID for recharge lag calculation
    GWLOB = None

# Derived input
    # Near Surface Soil water Storage [mm]
    SURFSTOR = None
    # Moisture Holding capacity by Soil type, should be loaded after loading SOL and
      SoilTab
    FRACSTOR = None
    # Tension Height of Soil [m]
    SoilTHT = None
    # Soil water content at the saturation point [mm]
    SoilSAT = None

# Output
    '''
    INT = None #Daily Interception [mm/day]
    ROF = None #Daily Direct Runoff [mm/day]
    EVopen = None #Daily Evaporation from the impervious or water surface [mm/day]
    PET = None #Daily Potential Evapotranspiration [mm/day]
    AET = None #Daily Actual Evapotranspiration [mm/day]
    '''
    #Actual recharge on i day [mm/day]
    RCHa = None
    #Soil Moisture Deficit at the end of the day [mm/day]
    SMD = None

# Parameter Tables
    #Land Cover table. It includes CN2o,Crop type, and area fraction.
    LcovTab = None
```

```
    # Soil Property table
    SoilTab = None
    # Crop Property table
    VegeTab = None #Crop Property table
    # Set of 1 year sequences for Kc_Tab, Kcb_Tab, CrH, Zr
    VegeGrowth = None

# Path list for Parameter Tables
    MnamFpath = None #MODFLOW nam file path

# Checking flags for Input files and Initializing class VELAS
    def __init__(self,MainFpath):
        self.MainFpath = MainFpath
        self.Ibound = IOman.getAscii2Array(MainFpath + IOF_init.IboundF, 'i')
        self.ws_zone = \
            IOman.getAscii2Array(MainFpath + IOF_init.IboundF, 'i', self.Ibound)
        self.ActiveArrayShape = self.ws_zone.shape
        self.numcells = len(self.ws_zone)
        self.DEM = IOman.getAscii2Array(MainFpath + IOF_init.DEMF, 'f',self.Ibound)
        self.SLP = IOman.getAscii2Array(MainFpath + IOF_init.SlopeF, 'f',self.Ibound)
        self.SOL = IOman.getAscii2Array(MainFpath + IOF_init.SoilF, 'i',self.Ibound)
        self.rLat = IOman.getAscii2Array(MainFpath + IOF_init.rLatF, 'f',self.Ibound)

        self.LcovTab = self.load_LcovTable(MainFpath + IOF_init.LCparamF)
        self.SoilTab = self.load_SoilTable(MainFpath + IOF_init.SoilparamF)
        self.SoilSAT,self.FRACSTOR,self.SoilTHT = \
            self.set_SAT_FRAC_THT(self.SOL, self.SoilTab)
        self.VegeTab = self.load_VegeTable(MainFpath + IOF_init.VegeparmF)
        self.VegeGrowth = self.set_VegeGrowth(self.VegeTab, self.Ke)
        self.RCHa = zeros(self.ActiveArrayShape,'float')
        self.SURFSTOR = zeros(self.ActiveArrayShape,'float')

        self.inPRCPpath = MainFpath + IOF_init.PRCP
        self.inTAVGpath = MainFpath + IOF_init.TAVG
        self.inTMINpath = MainFpath + IOF_init.TMIN
        self.inTMAXpath = MainFpath + IOF_init.TMAX
        self.inRHpath = MainFpath + IOF_init.RH
        self.inWINDpath = MainFpath + IOF_init.WIND
        self.inLCVpath = MainFpath + IOF_init.LCV
        self.inVEGpath = MainFpath + IOF_init.VEG

        self.outERRpath = MainFpath + IOF_init.ERR
        self.outEVTpath = MainFpath + IOF_init.EVT
        self.outINTpath = MainFpath + IOF_init.INT
        self.outROFpath = MainFpath + IOF_init.ROF
        self.outRCHpath = MainFpath + IOF_init.RCH
        self.outSMDpath = MainFpath + IOF_init.SMD

        self.outSMCpath = MainFpath + IOF_init.SMC
```

```
        self.outTAWpath = MainFpath + IOF_init.TAW
        self.outRAWpath = MainFpath + IOF_init.RAW
        self.outPETpath = MainFpath + IOF_init.PET
        self.outPERCpath = MainFpath + IOF_init.PERC

# Functions for parameter table loading
    def load_ParamTable(self,Fpath,Type):
        """
        A general form of parameter table loading function
        """
        ParamFile = csv.reader(open(Fpath))
        headers = ParamFile.next()
        ParamTab = {}
        for row in ParamFile:
            ParamTab[int(row[0])] = \
                dict(zip(headers[1:],row[1:3] + [float(x) for x in row[3:]]))

        if Type == 'LC':
            self.LcovTab = ParamTab
        elif Type == 'Soil':
            self.SoilTab = ParamTab
        else:
            self.VegeTab = ParamTab

    def load_LcovTable(self,LcovFpath):
        """
        Land cover property table loader
            It includes CN2o, Crop type, and area fraction

        default property header form
            index,LC,Land_Type,index_NGS,A,B,C,D,aVege_Soil,aOther
            (case sensitive)

        Usage
            LcovTab[LC index][property name]
        """
        LcovFile = csv.reader(open(LcovFpath))
        headers = LcovFile.next()
        LcovTab = {}
        for row in LcovFile:
            LcovTab[int(row[0])] = \
                dict(zip(headers[1:],row[1:3] + [float(x) for x in row[3:]]))
        return LcovTab

    def load_SoilTable(self,SoilFpath):
        """
        Soil property table Loader

        default property header form
```

```
            index,Soil_Texture,HSG,SAT,FC,WP,REW,TEW,FRACSTOR,Ze
            (case sensitive)

        Usage
            SoilTab[soil index][property name]
        """
        SoilFile = csv.reader(open(SoilFpath))
        headers = SoilFile.next()
        SoilTab = {}
        for row in SoilFile:
            SoilTab[int(row[0])] = \
                dict(zip(headers[1:],row[1:3] + [float(x) for x in row[3:]]))

        return SoilTab

    def set_SAT_FRAC_THT(self,SOL,SoilTab):
        '''
        function to create a grid containing spatial FRACSTOR distribution
        '''
        SAT = []
        FRACSTOR = []
        TENSHT = []

        for idx in SOL:
            SAT.append(SoilTab[idx]['SAT'])
            FRACSTOR.append(SoilTab[idx]['FRACSTOR'])
            TENSHT.append(SoilTab[idx]['TENSHT'])

        SAT = array(SAT)
        FRACSTOR = array(FRACSTOR)
        TENSHT = array(TENSHT)
        return SAT,FRACSTOR,TENSHT

    def load_VegeTable(self,VegeFpath):
        """
        Vegetation property table Loader

        default property header form
            index,Vege_Name,Vege_Type,L_ini,L_dev,L_mid,L_late,Plant_Date,
Kc_ini,Kc_mid,Kc_end,Kcb_ini,Kcb_mid,Kcb_end,HT_max,Root_Depth,p,aVEG_min,aVEG_max
            LAI_min,LAI_max
            (case sensitive)

        Usage
            VegeTab[Crop index][property name]
        """
        CropFile = csv.reader(open(VegeFpath))
        headers = CropFile.next()
```

```
        VegeTab = {}
        for row in CropFile:
            VegeTab[int(row[0])] = \
                dict(zip(headers[1:],row[1:3] + [float(x) for x in row[3:]]))

        return VegeTab

    def set_VegeGrowth(self,VegeTab,Ke):
        """
        Vegetation Growth Planner
            This function set daily crop coefficient for each crop for one full year.
            requirement: VegeTab (self.VegeTab)

        Usage
            self.VegeGrowth[Vege index][param name][iDay]
            self.VegeGrowth[30]['Kc_Tab'][35] -> it returns Kc_Tab value of crop index
30 on 35th day of the year
        """
        VegeGrowth = {}
        VegeGrowth_Keys = ['Kc_Tab','VegeHeight','RootDepth','aVEG','LAI']

        for index in VegeTab.keys(): #for every crop in the table
            if VegeTab[index]['Vege_Type'] == 'M_Tree': #for Mixed forest
                for idx in VegeTab.keys(): #Finding index number of conifer tree
                    if VegeTab[idx]['Vege_Type'] == 'C_Tree':
                        Conifer_index = idx
                        Kc_conifer = VegeTab[Conifer_index]['Kc_ini']
                        VegeGrowth[index] = \
dict(zip(VegeGrowth_Keys,SceneGen.get_VeGrSeries(Ke,VegeTab[index],Kc_conifer)))
            else:
                VegeGrowth[index] =  \
dict(zip(VegeGrowth_Keys,SceneGen.get_VeGrSeries(Ke,VegeTab[index])))

        return VegeGrowth

    def get_iDayParams(self,iDay):
        '''
        Returns required parameters on ith day of the year

        iDay: ith day of the year
        '''
        gKc_Tab = []; gCrH = []; gCN2o = [];
        gaVege = []; gaSoil = []; gaOther = []; gLAI = [];
        gTAW = []; gRAW = []; gcKc = []; gSMC_SAT = []; gSMC_FC = []; gSMC_WP = [];
        giZ = []

        for cell_id in range(self.numcells):
            VegeIdx = self.VEG[cell_id]
            LcovIdx = self.LCV[cell_id]
```

```
            SoilIdx = self.SOL[cell_id]
            Tmin = self.Tmin[cell_id]
            Tmax = self.Tmax[cell_id];

            # Windspeed Raster
            if self.u2 != None:
                u2 = self.u2[cell_id]

            # Required parameters
            iKc_Tab = self.VegeGrowth[VegeIdx]['Kc_Tab'][iDay]
            iCrH = self.VegeGrowth[VegeIdx]['VegeHeight'][iDay]
            iZr = self.VegeGrowth[VegeIdx]['RootDepth'][iDay]
            iaVEG = self.VegeGrowth[VegeIdx]['aVEG'][iDay]
            iLAI = self.VegeGrowth[VegeIdx]['LAI'][iDay]
            p = self.VegeTab[VegeIdx]['p'] #Depletion Fraction
            LCProtsGS = self.LcovTab[LcovIdx]
            LCProtsNGS = self.LcovTab[LCProtsGS['index_NGS']]
            SProts = self.SoilTab[SoilIdx]

            #add LAI values to grid
            gLAI.append(iLAI)

            #add cell Kc and CrH to grid
            iKc_Tab = \
                EVTLIB.get_Kcadj(iKc_Tab, Tmax, Tmin, u2, iCrH) #Climate adjustment
            gKc_Tab.append(iKc_Tab)
            gCrH.append(iCrH)

            #add CN2o and aFrac to grid
            iCN2o, iaFrac = \
                SceneGen.get_iCN2o_aFrac(iaVEG,LCProtsGS,LCProtsNGS,SProts)
            gCN2o.append(iCN2o)
            gaVege.append(iaFrac['Vege'])
            gaSoil.append(iaFrac['Soil'])
            gaOther.append(iaFrac['Other'])

            #Daily SMD parameters
            iTAW,iRAW,icKc,iSMC_SAT,iSMC_FC,iSMC_WP,iZ = \
                SceneGen.get_iSMDparams(iaVEG,iZr,iKc_Tab,p,self.Ke,SProts,LCProtsGS)
            gTAW.append(iTAW)
            gRAW.append(iRAW)
            gcKc.append(icKc)
            gSMC_SAT.append(iSMC_SAT)
            gSMC_FC.append(iSMC_FC)
            gSMC_WP.append(iSMC_WP)
            giZ.append(iZ)

        #List to Array conversion
        gKc_Tab = array(gKc_Tab) #Kc_Tab values [-]
```

```
    gcKc_Tab = array(gcKc) #Combined (Kc_Tab and Ke)Kc values [-]
    gLAI = array(gLAI) #LAI for the interception calculation
    gCrH = array(gCrH) #Crop (Vegetation) Height on the iDay [m]
    gCN2o = array(gCN2o) #Area weighted Curve Number [-]
    gaFrac = {'Vege':array(gaVege),'Soil':array(gaSoil),'Other':array(gaOther)}
           # Actual Area Fraction [-]
    gTAW = array(gTAW) #Total Available Water [-]
    gRAW = array(gRAW) #Readily Available Water [-]
    gSMC_SAT = array(gSMC_SAT) #Soil Moisture Content at the saturation point [mm]
    gSMC_FC = array(gSMC_FC) #Soil Moisture Content at the field capacity [mm]
    gSMC_WP = array(gSMC_WP) #Soil Moisture Content at the wilting point [mm]
    giZ = array(giZ) #Rootdepth or EV depth [m]

    # Groundwater Raster
    if self.GWL != None:
       GWdepth = self.DEM - (self.GWL + self.SoilTHT)
       GWS = where(giZ > GWdepth, self.SoilSAT * (giZ - GWdepth) * 1000.0, 0.0)
    else:
       GWS = zeros(self.ActiveArrayShape,'float')

    # Apply area fraction
    perviousArea = gaFrac['Vege'] + gaFrac['Soil']
    gTAW = gTAW * perviousArea
    gRAW = gRAW * perviousArea
    gSMC_SAT = gSMC_SAT * perviousArea
    gSMC_FC = gSMC_FC * perviousArea
    gSMC_WP = gSMC_WP * perviousArea
    GWS = GWS * perviousArea

    return gcKc_Tab,gLAI,gCN2o,gaFrac,gTAW,gRAW,gSMC_SAT,gSMC_FC,gSMC_WP,GWS

def load_iYearInputs(self,iYear):
    '''
    Input file loader

    iYear: ith year of the modeling period
    '''
    year = '\\' + str(iYear)
    PRCPfiles = os.listdir(self.inPRCPpath + year)
    TAVGfiles = os.listdir(self.inTAVGpath + year)
    TMINfiles = os.listdir(self.inTMINpath + year)
    TMAXfiles = os.listdir(self.inTMAXpath + year)
    RHfiles = os.listdir(self.inRHpath + year)

    WINDpath = self.inWINDpath + year
    if os.path.exists(WINDpath):
       WINDfiles = os.listdir(WINDpath)
    else:
       WINDpath = self.inWINDpath + IOF_init.WINDrefYear
```

```
         if os.path.exists(WINDpath):
            WINDfiles = os.listdir(WINDpath)
         else:
            WINDfiles = None

      self.LCV = IOman.getAscii2Array(self.inLCVpath + IOF_init.LCVprefix +
str(iYear) + '.asc', 'i',self.Ibound)
      self.VEG = IOman.getAscii2Array(self.inVEGpath + IOF_init.VEGprefix +
str(iYear) + '.asc', 'i',self.Ibound)

      return PRCPfiles,TAVGfiles,TMINfiles,TMAXfiles,RHfiles,WINDfiles

   def set_iDayInputs(self,year,iDay,PRCPfiles,TAVGfiles,\
                      TMINfiles,TMAXfiles,RHfiles,WINDfiles):
      '''
      Assigning iDay's input

      iDay: ith day of the year
      '''
      iDay = iDay - 1
      year = '\\' + str(year) + '\\'
      self.PPT = self.zero_fill(IOman.getAscii2Array(self.inPRCPpath + year +
PRCPfiles[iDay], 'f',self.Ibound))
      self.Tavg = IOman.getAscii2Array(self.inTAVGpath + year + TAVGfiles[iDay],
'f',self.Ibound)
      self.Tmin = IOman.getAscii2Array(self.inTMINpath + year + TMINfiles[iDay],
'f',self.Ibound)
      self.Tmax = IOman.getAscii2Array(self.inTMAXpath + year + TMAXfiles[iDay],
'f',self.Ibound)
      self.RHmean = self.zero_fill(IOman.getAscii2Array(self.inRHpath + year +
RHfiles[iDay], 'f',self.Ibound))

      if WINDfiles != None:
         WINDpath = self.inWINDpath + year + WINDfiles[iDay]
         if os.path.exists(WINDpath):
            self.u2 = self.zero_fill(IOman.getAscii2Array(WINDpath,
'f',self.Ibound))
         else:
            WINDpath = self.inWINDpath + IOF_init.WINDrefYear + '\\' +
WINDfiles[iDay]
            self.u2 = self.zero_fill(IOman.getAscii2Array(WINDpath,
'f',self.Ibound))
      else:
         self.u2 = None

   def zero_fill(self,dArray):
      dArray = where(dArray < 0.0, 0.0, dArray)
      return dArray
```

```
   def Save_iDayOutputs(self,year,iDay,error,INT,ROF,\
                        AET,RCH,SMD,PERC,PET,TAW,RAW,SMC):
      '''
      Results save function
      '''
      ESRI_Header = IOF_init.ESRI_Header
      year = str(year); iDay = str(iDay);

      errorF = self.outERRpath + '\\' + year; IOman.make_folder(errorF); errorF =
errorF + '\\' + IOman.Fname_PYD('ERR', year, iDay);
      IOman.Save_Array_as_ASCII(ESRI_Header,self.Ibound,error,errorF)

      INTF = self.outINTpath + '\\' + year; IOman.make_folder(INTF); INTF = INTF +
'\\' + IOman.Fname_PYD('INT', year, iDay);
      IOman.Save_Array_as_ASCII(ESRI_Header,self.Ibound,INT,INTF)

      ROFF = self.outROFpath + '\\' + year; IOman.make_folder(ROFF); ROFF = ROFF +
'\\' + IOman.Fname_PYD('ROF', year, iDay);
      IOman.Save_Array_as_ASCII(ESRI_Header,self.Ibound,ROF,ROFF)

      AETF = self.outEVTpath + '\\' + year; IOman.make_folder(AETF); AETF = AETF +
'\\' + IOman.Fname_PYD('EVT', year, iDay);
      IOman.Save_Array_as_ASCII(ESRI_Header,self.Ibound,AET,AETF)

      RCHF = self.outRCHpath + '\\' + year; IOman.make_folder(RCHF); RCHF = RCHF +
'\\' + IOman.Fname_PYD('RCH', year, iDay);
      IOman.Save_Array_as_ASCII(ESRI_Header,self.Ibound,RCH,RCHF)

      SMDF = self.outSMDpath + '\\' + year; IOman.make_folder(SMDF); SMDF = SMDF +
'\\' + IOman.Fname_PYD('SMD', year, iDay);
      IOman.Save_Array_as_ASCII(ESRI_Header,self.Ibound,SMD,SMDF)

      PERCF = self.outPERCpath + '\\' + year; IOman.make_folder(PERCF); PERCF =
PERCF + '\\' + IOman.Fname_PYD('PERC', year, iDay);
      IOman.Save_Array_as_ASCII(ESRI_Header,self.Ibound,PERC,PERCF)

      PETF = self.outPETpath + '\\' + year; IOman.make_folder(PETF); PETF = PETF +
'\\' + IOman.Fname_PYD('PET', year, iDay);
      IOman.Save_Array_as_ASCII(ESRI_Header,self.Ibound,PET,PETF)

      TAWF = self.outTAWpath + '\\' + year; IOman.make_folder(TAWF); TAWF = TAWF +
'\\' + IOman.Fname_PYD('TAW', year, iDay);
      IOman.Save_Array_as_ASCII(ESRI_Header,self.Ibound,TAW,TAWF)

      RAWF = self.outRAWpath + '\\' + year; IOman.make_folder(RAWF); RAWF = RAWF +
'\\' + IOman.Fname_PYD('RAW', year, iDay);
      IOman.Save_Array_as_ASCII(ESRI_Header,self.Ibound,RAW,RAWF)
```

```
      SMCF = self.outSMCpath + '\\' + year; IOman.make_folder(SMCF); SMCF = SMCF +
'\\' + IOman.Fname_PYD('SMC', year, iDay);
      IOman.Save_Array_as_ASCII(ESRI_Header,self.Ibound,SMC,SMCF)

   def call_initSM(self,year):
      if isleap(year - 1):
         endDay = '366'
      else:
         endDay = '365'

      SMDoPath = self.outSMDpath + '\\' + str(year - 1) + '\\' +
IOman.Fname_PYD('SMD', str(year - 1), endDay)
      SMDo = IOman.getAscii2Array(SMDoPath, 'f',self.Ibound)
      print IOman.Fname_PYD('SMD', str(year - 1), endDay), '... loaded as initial'
      SMCoPath = self.outSMCpath + '\\' + str(year - 1) + '\\' +
IOman.Fname_PYD('SMC', str(year - 1), endDay)
      SMCo = IOman.getAscii2Array(SMCoPath, 'f',self.Ibound)
      print IOman.Fname_PYD('SMC', str(year - 1), endDay), '... loaded as initial'

      return SMDo, SMCo

   def get_iINT(self,PPT,iLAI,iaFrac):
      '''
      function for calculating the interception on the ith day

      input
         iLAI: LAI on the ith day of the year [-]
         iaFrac: Area fraction data to weight the area of vegetation
               {array(aVege),array(aSoil),array(aOther)}

      output
         INT: the interception on the ith day [mm]
         PPT: the precipitation after the interception [mm]
      '''
      INT = RainINT.cal_Interception(PPT,iLAI)
      INT = INT * iaFrac['Vege']
      iPPT = PPT - INT

      return INT, iPPT

   def get_iROF(self,PPT,iCN2o,IaS_ratio,Slope,SMC,SMC_WP,SMC_FC,SMC_SAT):
      '''
      function for calculating the runoff on the ith day

      input
         PPT: precipitation after the interception [mm]
         iCN2o: area weighted curve numbers before the adjustment by slope and SMC
         IaS_ratio: NRCS CN method Ia/S value: 0.2 for original or 0.05 for better
fit
```

```
            Slope: slope grid [-] (not percent or degree)
            SMC: current soil moisture content [mm]
            SMC_WP: Soil Moisture Content at wilting point [mm]
            SMC_FC: Soil Moisture content at field capacity [mm]
            SMC_SAT: Soil Moisture Content at saturation point [mm]

        output
            ROF: Runoff on the iDay of the year [mm]
            INF: Infiltration from surface (P-I-ROF) [mm]
        '''

        ROF =
NRCSCNrunoff.cal_Runoff(PPT,iCN2o,IaS_ratio,Slope,SMC,SMC_WP,SMC_FC,SMC_SAT)
        return ROF

    def get_iPET(self,iDay,icKc,equation = 'FAO56-PM'):
        '''
        function for calculating the potential evapotranspiration on the iDay
        FAO56-PM equation is default
        '''
        if self.u2 == None:
            ETo =
EVTLIB.cal_EToVa(self.Tavg,self.Tmin,self.Tmax,self.RHmean,self.u2,self.DEM,self.rLat
,iDay,self.kRs)
        else:
            if equation != 'FAO56-PM': #by the simplified version of the Penman method
                ETo =
EVTLIB.cal_EToVa(self.Tavg,self.Tmin,self.Tmax,self.RHmean,self.u2,self.DEM,self.rLat
,iDay,self.kRs)
            else: #by the FAO56-PM method
                ETo =
EVTLIB.cal_EToPM(self.Tmax,self.Tmin,self.Tavg,self.RHmean,self.u2,self.DEM,self.rLat
,iDay,self.kRs)

        iPET = icKc * ETo
        return iPET

    def get_iEopen(self,iDay,equation = 'Penman'):
        if self.u2 == None:
            Eopen = EVTLIB.cal_EVopenVa(self.Tavg, self.Tmin, self.Tmax, self.RHmean,
self.u2, self.DEM, self.rLat, iDay, self.kRs)
        else:
            if equation != 'Penman':
                Eopen = EVTLIB.cal_EVopenVa(self.Tavg, self.Tmin, self.Tmax, self.RHmean,
self.u2, self.DEM, self.rLat, iDay, self.kRs)
            else:
                Eopen = EVTLIB.cal_EVpen(self.Tavg, self.Tmin, self.Tmax, self.RHmean,
self.u2, self.DEM, self.rLat, iDay, self.kRs)
```

```
        return Eopen

    def get_iSWB(self,EVopen,sINF,PET,SURFSTORo,FRACSTOR,iTAW,iRAW,SMDo,GWS):

        AET,SURFSTOR,SMD,PERC =
ISWB.cal_gNSSWB(self.LCV,self.LcovTab,EVopen,sINF,PET,SURFSTORo,FRACSTOR,iTAW,iRAW,SM
Do,GWS)

        return AET,SURFSTOR,SMD,PERC

    def cal_error(self,PPT,INT,ROF,sINF,AET,SURFSTOR,SMDo,SMD,PERC,EVopen):
        DELTA_SMD = SMDo - SMD

        SURFSTORo = where(PPT < sINF, sINF - PPT + INT + ROF, 0.0)

        error = PPT + SURFSTORo - INT - ROF - AET - SURFSTOR - PERC - DELTA_SMD

        for i in range(len(self.LCV)):
            if self.LcovTab[self.LCV[i]]['Land_Type'] == 'Water':
                error[i] = PPT[i] - INT[i] - ROF[i]

        return error, AET

    def iyrRun(self,year, Run = True):
        '''
        Yearly simulation
        Run = True (normal run), False (initial run)
        '''

        # leap year handling
        if isleap(year):
            endDay = 367
        else:
            endDay = 366

         PRCPfiles,TAVGfiles,TMINfiles,TMAXfiles,RHfiles,WINDfiles =
self.load_iYearInputs(year)

        if Run == True:
            if self.SMD != None:
                print 'Initial SMD loaded from memory...'
            else:
                self.SMD, iSMC = self.call_initSM(year)
                print 'Initial SMD loaded from file'
        else:
            if self.SMD == None:
                print 'SMD initialized with default setting'
            else:
                self.SMD = None
```

```
            print 'SMD initialized with default setting'
        # Create MODFLOW instance if GWmodel exists
        if self.GWmodel == True:
            GWMOD = MODFdriver.MODFLOW(self.MexeFPath, self.MnamFPath, self.numcell)
        # Daily loop
        for iDay in range(1,endDay):
            self.set_iDayInputs(year, iDay, PRCPfiles, TAVGfiles, TMINfiles, TMAXfiles,
RHfiles, WINDfiles)

            icKc,iLAI,iCN2o,iaFrac,iTAW,iRAW,iSMC_SAT,iSMC_FC,iSMC_WP,GWS =
self.get_iDayParams(iDay)

            #A-Weighted INT and PPT after INT
            INT, iPPT = self.get_iINT(self.PPT,iLAI,iaFrac)

            # Soil Moisture Content for Runoff calculation
            if self.SMD == None:
                iSMC = (iSMC_FC + iSMC_WP) * 0.5
                self.SMD = iSMC_FC - iSMC
            else:
                iSMC = iSMC_FC - self.SMD + GWS

            #NRCS Runoff
            ROF =
self.get_iROF(iPPT,iCN2o,self.IaS_ratio,self.SLP,iSMC,iSMC_WP,iSMC_FC,iSMC_SAT)
            sINF = iPPT - ROF

            # open water evaporation
            EVopen = self.get_iEopen(iDay) * iaFrac['Other']

            # Potential Evapotranspiration - vegetated and soil surface
            perviousArea = iaFrac['Vege'] + iaFrac['Soil']
            PET = self.get_iPET(iDay,icKc) * perviousArea

            # PET correction for impervious cell
            iPET = PET + EVopen
            #Soil Moisture Balance
            SMDo = copy(self.SMD)
            SURFSTORo = copy(self.SURFSTOR)
            AET,self.SURFSTOR,self.SMD,PERC =
self.get_iSWB(EVopen,sINF,PET,SURFSTORo,self.FRACSTOR,iTAW,iRAW,SMDo,GWS)
            error, AET =
self.cal_error(self.PPT,INT,ROF,sINF,AET,self.SURFSTOR,SMDo,self.SMD,PERC,EVopen)

            #Actual Recharge
            self.RCHa = ActualRCH.cal_aRCH(PERC,self.RCHa,self.deltaGW)

            #Run MODFLOW
            if self.GWmodel == True:
```

```
    self.GWL = GWMOD.run(self.RCHa)

# SMC today
iSMCtoday = iSMC_FC - self.SMD + GWS

# Save outputs of iDay
if Run == True:
    self.Save_iDayOutputs(year,iDay,error,INT,ROF,AET,\
                          self.RCHa,self.SMD,PERC,iPET,iTAW,iRAW,iSMCtoday)
    print str(year) + ' ' + str(iDay) + ' Calcualtion Done!'
else:
    print str(year) + ' ' + str(iDay) + ' Calcualtion Done!'
```

```
#===============================================================================
# SceneGen
#
#  supporting Library for Crop Growth and Land cover transition
#
#    Python 2.5 code by Changhui Park <Changhui.Park@umkc.edu>
#    Date 07.20.2011
#
# Hydro Lab.
# Department of Geosciences, University of Missouri - Kansas City
#===============================================================================

def get_iCN2o_aFrac(iaVEG,LCProtsGS,LCProtsNGS,SProts):
    '''
   Returns the area weighted CN2o and the actual area fractions of a cell on ith day
of the year
      (aVege + aSoil + aOther = 1.0) aOther is water or impervious surface

   iaVEG: area fraction of vegetation on ith day of the year
   LCProtsGS: Land cover Info from LcovTab (Growing Season)
   LCProtsNGS: Land cover Info from LcovTab (Non-Growing Season)
   SProts: Soil Info from SoilTab
    '''
   Land_Type = LCProtsGS['Land_Type']

   #actual area fractions of a cell
   aVege = LCProtsGS['aVege_Soil'] * iaVEG
   aSoil = LCProtsGS['aVege_Soil'] - aVege
   aOther = LCProtsGS['aOther']

   #Bare soil handling : 0 fraction for vege and other
   if Land_Type == 'Bare':
      aVege = 0.0
      aSoil = 1.0
      aOther = 0.0

   HSG = SProts['HSG']

   if Land_Type == 'Water' or Land_Type == 'Bare': #in case of water or bare surface
as a land cover type
      CN2o = LCProtsGS[HSG]
   else:#area weighting
      CN2o_GS = LCProtsGS[HSG]
      CN2o_NGS = LCProtsNGS[HSG]
      CN2o = CN2o_GS * (aVege + aOther) + CN2o_NGS * aSoil

   aFrac = {'Vege':aVege,'Soil':aSoil,'Other':aOther}
   return CN2o,aFrac
```

```
def get_iSMDparams(iaVEG,iZr,iKc_Tab,p,Ke,SProts,LCProtsGS):
    '''
    Returns combined (area weighted) TAW, RAW, and Kc_Tab on ith day of the year

    iaVEG: area fractions of Vegetation on ith day of the year
    VProts: Vege Info from VegeTab
    SProts: Soil Info from SoilTab
    iZr: Root Depth on ith day of the year
    iKc: Kc_Tab on ith day of the year
    Ke: Soil Evaporation coefficient from Main module
    '''
    #Bare soil handling; aSoil is always 1.0
    Land_Type = LCProtsGS['Land_Type']
    if Land_Type == 'Bare':
        aVege = 0.0
        aSoil = 1.0
        iZr = 0.0
    else:
        aVege = iaVEG
        aSoil = 1.0 - aVege
    #-------------------------------------------------
    SAT = SProts['SAT']
    FC = SProts['FC']
    WP = SProts['WP']
    Ze = SProts['Ze']

    PAWmax = 1000.0 * (FC - WP) * iZr
    PAW = p * PAWmax
    TEW = SProts['TEW']
    REW = SProts['REW']

    TAW = aVege * PAWmax + aSoil * TEW
    RAW = aVege * PAW + aSoil * REW
    cKc = aVege * iKc_Tab + aSoil * Ke

    if iZr < Ze:
        iZ = Ze
    else:
        iZ = aVege * iZr + aSoil * Ze

    SMC_SAT = SAT * iZ * 1000.0
    SMC_FC = FC * iZ * 1000.0
    SMC_WP = WP * iZ * 1000.0

    return TAW,RAW,cKc,SMC_SAT,SMC_FC,SMC_WP,iZ

def get_VeGrSeries(Ke,VProts,Kc_conifer = None):
    """
```

```
   Daily Crop Coefficient Kc_Tab, Kcb_Tab, VegeHeight, and RootDepth generator for 1
full year
   This function is called by set_VegeGrowth() in Main module

   Ke: Soil Evaporation coefficient from Main module
   VProts: Vege Info from VegeTab
   Kc_conifer: Kc_ini value of conifer tree for Kc values of mixed forest in non-
growing season
   Kcb_conifer: Kcb_ini value of conifer tree for Kcb values of mixed fores in non-
growing season
   """
   VegeType = VProts['Vege_Type']

   if VegeType == 'Crop':
      return get_CropGrowth(Ke,VProts)
   elif VegeType == 'C_Tree':
      return get_CTreeGrowth(Ke,VProts)
   elif VegeType == 'D_Tree':
      return get_DMTreeGrowth(Ke,VProts)
   elif VegeType == 'M_Tree':
      return get_DMTreeGrowth(Ke,VProts,Kc_conifer)
   else:
      return get_GrassGrowth(Ke,VProts)

def get_CropGrowth(Ke,VProts):
   #Loading parameters
   L_ini = int(VProts['L_ini']); L_dev = int(VProts['L_dev']); L_mid =
int(VProts['L_mid']); L_late = int(VProts['L_late']);
   PlantDate = int(VProts['Plant_Date'])
   Init2Dev = L_ini + L_dev; Init2Mid = Init2Dev + L_mid; HarvestDate = PlantDate +
L_ini + L_dev + L_mid + L_late;
   Kc_ini = VProts['Kc_ini']; Kc_mid = VProts['Kc_mid']; Kc_end = VProts['Kc_end'];
   HTmax = VProts['HT_max']; Zrmax = VProts['Zr_max'];
   aVEG_min = VProts['aVEG_min']; aVEG_max = VProts['aVEG_max']; LAI_min =
VProts['LAI_min']; LAI_max = VProts['LAI_max'];

   #Base values for the year
   Kc_List = [Ke] * 367
   CrH_List = [0.] * 367
   Zr_List = [0.] * 367
   aVEG_List = [aVEG_min] * 367
   LAI_List = [LAI_min] * 367

   #Kc values
   Kc_Series = \
            [Kc_ini] * L_ini +
[get_Kci(Kc_ini,Kc_mid,L_ini,L_dev,L_mid,L_ini+igDay+1.0) for igDay in
range(int(L_dev))] \
```

```
          + [Kc_mid] * L_mid +
[get_Kci(Kc_ini,Kc_mid,L_ini,L_dev,L_mid,Init2Mid+igDay+1.0,Kc_end,L_late) for igDay
in range(int(L_late))]

   #Crop Height
   CrH_d1 = HTmax /(Init2Dev + 1)
   CrH_Series = [CrH_d1 + CrH_d1 * iDay for iDay in range(Init2Dev)] + [HTmax] *
(L_mid + L_late)

   #Root Depth
   Zr_d1 = Zrmax / (Init2Dev + 1)
   Zr_Series =  [Zr_d1 + Zr_d1 * iDay for iDay in range(Init2Dev)] + [Zrmax] * (L_mid
+ L_late)

   #Vege area fraction
   aVEG = aVEG_max - aVEG_min; aVEG_d1 = aVEG / (Init2Dev + 1)
   aVEG_Series = [aVEG_min + aVEG_d1 + aVEG_d1 * iDay for iDay in range(Init2Dev)] +
[aVEG_max] * (L_mid + L_late)

   #LAI
   LAI = LAI_max - LAI_min; LAI_d1 = LAI / (Init2Dev + 1)
   LAI_Series = [LAI_min + LAI_d1 + LAI_d1 * iDay for iDay in range(Init2Dev)] +
[LAI_max] * (L_mid + L_late)

   #Mosaicking with base values
   if HarvestDate > 365: #for winter sown crop
      nDaysP1 = 366 - PlantDate
      nDaysP2 = HarvestDate - 366
      Kc_List[PlantDate:] = Kc_Series[:nDaysP1]; Kc_List[:nDaysP2+1] =
Kc_Series[nDaysP1-1:];
      CrH_List[PlantDate:] = CrH_Series[:nDaysP1]; CrH_List[:nDaysP2+1] =
CrH_Series[nDaysP1-1:];
      Zr_List[PlantDate:] = Zr_Series[:nDaysP1]; Zr_List[:nDaysP2+1] =
Zr_Series[nDaysP1-1:];
      aVEG_List[PlantDate:] = aVEG_Series[:nDaysP1]; aVEG_List[:nDaysP2+1] =
aVEG_Series[nDaysP1-1:];
      LAI_List[PlantDate:] = LAI_Series[:nDaysP1]; LAI_List[:nDaysP2+1] =
LAI_Series[nDaysP1-1:];
   else: #for spring sown crop
      Kc_List[PlantDate:HarvestDate] = Kc_Series
      CrH_List[PlantDate:HarvestDate] = CrH_Series
      Zr_List[PlantDate:HarvestDate] = Zr_Series
      aVEG_List[PlantDate:HarvestDate] = aVEG_Series
      LAI_List[PlantDate:HarvestDate] = LAI_Series

   return Kc_List,CrH_List,Zr_List,aVEG_List,LAI_List

def get_CTreeGrowth(Ke,VProts):
   #Loading parameters
```

```
   L_ini = int(VProts['L_ini']); L_dev = int(VProts['L_dev']); L_mid =
int(VProts['L_mid']); L_late = int(VProts['L_late']);
   WarmSeasonStart = int(VProts['Plant_Date'])
   Init2Dev = L_ini + L_dev; WarmSeasonEnd = WarmSeasonStart + L_ini + L_dev + L_mid
+ L_late;
   Kc_ini = VProts['Kc_ini']
   HTmax = VProts['HT_max']; Zrmax = VProts['Zr_max'];
   aVEG_min = VProts['aVEG_min']; aVEG_max = VProts['aVEG_max']; LAI_min =
VProts['LAI_min']; LAI_max = VProts['LAI_max'];

   #Kc values: constant for the year
   Kc_List = [Kc_ini] * 367

   #Vege Height: constant for the year
   CrH_List = [HTmax] * 367

   #Root Depth: constant for the year
   Zr_List = [Zrmax] * 367

   #Vege area fraction
   aVEG_List = [aVEG_min] * 367
   aVEG = aVEG_max - aVEG_min; aVEG_d1 = aVEG / (Init2Dev + 1); aVEG_d2 = -aVEG /
(L_late + 1);
   aVEG_Series = [aVEG_min + aVEG_d1 + aVEG_d1 * iDay for iDay in range(Init2Dev)] +
[aVEG_max] * L_mid \
             + [aVEG_max + aVEG_d2 + aVEG_d2 * iDay for iDay in range(L_late)]
   aVEG_List[WarmSeasonStart:WarmSeasonEnd] = aVEG_Series

   #LAI
   LAI_List = [LAI_min] * 367
   LAI = LAI_max - LAI_min; LAI_d1 = LAI / (Init2Dev + 1); LAI_d2 = -LAI / (L_late +
1);
   LAI_Series = [LAI_min + LAI_d1 + LAI_d1 * iDay for iDay in range(Init2Dev)] +
[LAI_max] * L_mid \
            + [LAI_max + LAI_d2 + LAI_d2 * iDay for iDay in range(L_late)]
   LAI_List[WarmSeasonStart:WarmSeasonEnd] = LAI_Series

   return Kc_List,CrH_List,Zr_List,aVEG_List,LAI_List

def get_DMTreeGrowth(Ke,VProts,Kc_conifer = None):
   #Loading parameters
   L_ini = int(VProts['L_ini']); L_dev = int(VProts['L_dev']); L_mid =
int(VProts['L_mid']); L_late = int(VProts['L_late']);
   WarmSeasonStart = int(VProts['Plant_Date'])
   Init2Dev = L_ini + L_dev; Init2Mid = Init2Dev + L_mid; WarmSeasonEnd =
WarmSeasonStart + L_ini + L_dev + L_mid + L_late;
   Kc_ini = VProts['Kc_ini']; Kc_mid = VProts['Kc_mid']; Kc_end = VProts['Kc_end'];
   HTmax = VProts['HT_max']; Zrmax = VProts['Zr_max'];
```

```
   aVEG_min = VProts['aVEG_min']; aVEG_max = VProts['aVEG_max']; LAI_min =
VProts['LAI_min']; LAI_max = VProts['LAI_max'];

   #Base values for the year
   if Kc_conifer != None: #for Mixed forest
      Kc_List = [Kc_conifer] * 367
   else:#for others than conifer and mixed
      Kc_List = [Kc_ini] * 367

   aVEG_List = [aVEG_min] * 367
   LAI_List = [LAI_min] * 367

   #Kc values
   Kc_Series = \
            [Kc_ini] * L_ini +
[get_Kci(Kc_ini,Kc_mid,L_ini,L_dev,L_mid,L_ini+igDay+1.0) for igDay in
range(int(L_dev))] \
          + [Kc_mid] * L_mid +
[get_Kci(Kc_ini,Kc_mid,L_ini,L_dev,L_mid,Init2Mid+igDay+1.0,Kc_end,L_late) for igDay
in range(int(L_late))]

   #Crop Height is constant for tree
   CrH_List = [HTmax] * 367

   #Root Depth is constant for tree
   Zr_List = [Zrmax] * 367

   #Vege area fraction
   aVEG = aVEG_max - aVEG_min; aVEG_d1 = aVEG / (Init2Dev + 1); aVEG_d2 = -aVEG /
(L_late + 1);
   aVEG_Series = [aVEG_min + aVEG_d1 + aVEG_d1 * iDay for iDay in range(Init2Dev)] +
[aVEG_max] * L_mid \
             + [aVEG_max + aVEG_d2 + aVEG_d2 * iDay for iDay in range(L_late)]

   #LAI
   LAI = LAI_max - LAI_min; LAI_d1 = LAI / (Init2Dev + 1); LAI_d2 = -LAI / (L_late +
1);
   LAI_Series = [LAI_min + LAI_d1 + LAI_d1 * iDay for iDay in range(Init2Dev)] +
[LAI_max] * L_mid \
            + [LAI_max + LAI_d2 + LAI_d2 * iDay for iDay in range(L_late)]

   Kc_List[WarmSeasonStart:WarmSeasonEnd] = Kc_Series
   aVEG_List[WarmSeasonStart:WarmSeasonEnd] = aVEG_Series
   LAI_List[WarmSeasonStart:WarmSeasonEnd] = LAI_Series

   return Kc_List,CrH_List,Zr_List,aVEG_List,LAI_List

def get_GrassGrowth(Ke,VProts):
   #Loading parameters
```

```
   L_ini = int(VProts['L_ini']); L_dev = int(VProts['L_dev']); L_mid =
int(VProts['L_mid']); L_late = int(VProts['L_late']);
   WarmSeasonStart = int(VProts['Plant_Date'])
   Init2Dev = L_ini + L_dev; Init2Mid = Init2Dev + L_mid; WarmSeasonEnd =
WarmSeasonStart + L_ini + L_dev + L_mid + L_late;
   Kc_ini = VProts['Kc_ini']; Kc_mid = VProts['Kc_mid']; Kc_end = VProts['Kc_end'];
   HTmax = VProts['HT_max']; Zrmax = VProts['Zr_max'];
   aVEG_min = VProts['aVEG_min']; aVEG_max = VProts['aVEG_max']; LAI_min =
VProts['LAI_min']; LAI_max = VProts['LAI_max'];

   #Base values for the year
   Kc_List = [Ke] * 367
   CrH_List = [0.] * 367
   Zr_List = [0.] * 367
   aVEG_List = [aVEG_min] * 367
   LAI_List = [LAI_min] * 367

   #Kc values
   Kc_Series = \
            [Kc_ini] * L_ini +
[get_Kci(Kc_ini,Kc_mid,L_ini,L_dev,L_mid,L_ini+igDay+1.0) for igDay in
range(int(L_dev))] \
          + [Kc_mid] * L_mid +
[get_Kci(Kc_ini,Kc_mid,L_ini,L_dev,L_mid,Init2Mid+igDay+1.0,Kc_end,L_late) for igDay
in range(int(L_late))]

   #Crop Height
   CrH_d1 = HTmax /(Init2Dev + 1)
   CrH_Series = [CrH_d1 + CrH_d1 * iDay for iDay in range(Init2Dev)] + [HTmax] *
(L_mid + L_late)

   #Root Depth
   Zr_d1 = Zrmax / (Init2Dev + 1)
   Zr_Series =  [Zr_d1 + Zr_d1 * iDay for iDay in range(Init2Dev)] + [Zrmax] * (L_mid
+ L_late)

   #Vege area fraction
   aVEG = aVEG_max - aVEG_min; aVEG_d1 = aVEG / (Init2Dev + 1); aVEG_d2 = -aVEG /
(L_late + 1);
   aVEG_Series = [aVEG_min + aVEG_d1 + aVEG_d1 * iDay for iDay in range(Init2Dev)] +
[aVEG_max] * L_mid \
             + [aVEG_max + aVEG_d2 + aVEG_d2 * iDay for iDay in range(L_late)]

   #LAI
   LAI = LAI_max - LAI_min; LAI_d1 = LAI / (Init2Dev + 1); LAI_d2 = -LAI / (L_late +
1);
   LAI_Series = [LAI_min + LAI_d1 + LAI_d1 * iDay for iDay in range(Init2Dev)] +
[LAI_max] * L_mid \
            + [LAI_max + LAI_d2 + LAI_d2 * iDay for iDay in range(L_late)]
```

```
    Kc_List[WarmSeasonStart:WarmSeasonEnd] = Kc_Series
    CrH_List[WarmSeasonStart:WarmSeasonEnd] = CrH_Series
    Zr_List[WarmSeasonStart:WarmSeasonEnd] = Zr_Series
    aVEG_List[WarmSeasonStart:WarmSeasonEnd] = aVEG_Series
    LAI_List[WarmSeasonStart:WarmSeasonEnd] = LAI_Series

    return Kc_List,CrH_List,Zr_List,aVEG_List,LAI_List

def get_Kci(Kc_ini,Kc_mid,L_ini,L_dev,L_mid,igDay,Kc_end=None,L_late=None):
    """
    Kc determination during crop development and late season using FAO56 Equation 66

    Input
        Kc_ini,Kc_mid,Kc_end(if needed): Kc for each growth stage taken from FAO56
Table 12 or 17
        L_ini,L_dev,L_mid,L_late(if needed): Lengths of crop development stages
        igDay: day number within the growing season (1 to length of growing season)
    Output
        Kc_i (dev): Kc for ith day during crop develelopment stage
        Kc_i (late): Kc for ith day during late stage (if Kc_end and L_late != None)
    """
    if Kc_end == None:
        Kc_i = Kc_ini + ((igDay - L_ini) / L_dev) * (Kc_mid - Kc_ini)
    else:
        Kc_i = Kc_mid + ((igDay - (L_ini + L_dev + L_mid)) / L_late) * (Kc_end -
Kc_mid)

    return Kc_i
```

```
#==============================================================================
# Input and Output manager
#
#   Python 2.5 code by Changhui Park <Changhui.Park@umkc.edu>
#   Date 07.20.2011
#
# Hydro Lab.
# Department of Geosciences, University of Missouri - Kansas City
#==============================================================================
from numpy import array,extract,nonzero
import os

def getAscii2Array(ASCIIfilePath,dFlag,Ibound = None):
    '''
    Loading ESRI ASCII grid

      input
        ASCIIfilePath: the full path of the Source ESRI ASCII file ' '
        dFlag: data type identifier ('i': integer, others: float) ' '
        Ibound: Active and Inactive cells information array() or None

      output
        data array from active cells array()
        in case of input ASCII is Ibound then it returns all cells (Active +
Inactive)
    '''
    # Open Input ESRI ASCII grid file
    ASCIIfile = open(ASCIIfilePath)

    # Read at once
    values = ASCIIfile.read()
    # by data type (integer or float)
    if dFlag == 'i':
        values = map(int, values.split()[12:]) #[12:] means stripping out headers
    else:
        values = map(float, values.split()[12:])

    if Ibound == None: # if input is Ibound
        return array(values) # return all cells
    else:
        return extract(Ibound,values) #return active cells only


def Save_Array_as_ASCII(ESRI_Header,Ibound,dArray,targetF):
    '''
    Save function
      input
         ESRI_Header: ESRI ASCII grid Header {}
         Ibound: Active and Inactive cells information array()
```

```
        dArray: dataset to be saved as ESRI ASCII grid
        targetF: the full path of ASCII file
    '''
    # Creating a container which is filled 'NODATA_value' and has the same size as
Ibound
    if isinstance(dArray.item(0),int):
        NoData = int(ESRI_Header['NODATA_value'])
    else:
        NoData = float(ESRI_Header['NODATA_value'])

    outArray = array([NoData] * len(Ibound))

    # Replacing NODATA with data
    outArray.put(nonzero(Ibound),dArray)
    # Reshaping for x,y matrix
    outArray.shape = (ESRI_Header['nrows'],ESRI_Header['ncols'])
    # open targetfile
    targetFile = open(targetF,'w')
    # Write header
    targetFile.write('ncols ' + str(ESRI_Header['ncols']) + '\n')
    targetFile.write('nrows ' + str(ESRI_Header['nrows']) + '\n')
    targetFile.write('xllcorner ' + str(ESRI_Header['xllcorner']) + '\n')
    targetFile.write('yllcorner ' + str(ESRI_Header['yllcorner']) + '\n')
    targetFile.write('cellsize ' + str(ESRI_Header['cellsize']) + '\n')
    targetFile.write('NODAT_value ' + str(NoData) + '\n')

    # Write all data
    for row in outArray:
        strRow = ' '.join(map(str,row))
        targetFile.write(strRow + '\n')

    # Close file
    targetFile.close()

def make_folder(path):
    if os.path.exists(path):
        pass
    else:
        os.makedirs(path)

def Fname_PYD(Prefix,year,iDay):
    if len(iDay) == 1:
        iDay = '_00' + iDay
    elif len(iDay) == 2:
        iDay = '_0' + iDay
    else:
        iDay = '_' + iDay

    return Prefix + '_' + year + iDay + '.asc'
```

```
#==============================================================================
# Input and Output file path definition file
#
#   Python 2.5 code by Changhui Park <Changhui.Park@umkc.edu>
#   Date 07.20.2011
#
# Hydro Lab.
# Department of Geosciences, University of Missouri - Kansas City
#==============================================================================
'''
   Input and output file paths should be specified in this file.
'''
# Input location
PRCP = '\\input\\prcp'
LCV = '\\input\\LCV'
LCVprefix = '\\LCV_'
VEG = '\\input\\VEG'
VEGprefix = '\\VEG_'
TAVG = '\\input\\tavg'
TMAX = '\\input\\tmax'
TMIN = '\\input\\tmin'
WIND = '\\input\\wind'
RH = '\\input\\RH'
COMM = '\\input\\Common'
PTab = '\\input\\ParamTables'

IboundF = COMM + '\\Ibound.asc'
SlopeF = COMM + '\\Slope.asc'
SoilF = COMM + '\\Soil.asc'
DEMF = COMM + '\\surface.asc'
rLatF = COMM + '\\radianLatitude.asc'

LCparamF = PTab + '\\LCproperties.csv'
SoilparamF = PTab + '\\SoilProperties.csv'
VegeparmF = PTab + '\\VegeProperties.csv'

# GWmodel
GWmodel = True
MexeFPath = '\\input\\GWMOD\\modflow.exe'
MnamFPath = '\\input\\GWMOD\\GWmodel.nam'

# Output location
EVT = '\\Output\\WBC\\Evapotranspiration'
INT = '\\Output\\WBC\\Interception'
RCH = '\\Output\\WBC\\Recharge'
ROF = '\\Output\\WBC\\Runoff'
SMD = '\\Output\\WBC\\SMD'
ERR = '\\Output\\WBC\\error'
```

```
# Scratch location
LAI = '\\Output\\xscratch\\LAI'
CNadj = '\\Output\\xscratch\\CN'
S_idx = '\\Output\\xscratch\\S_idx'
SMC = '\\Output\\xscratch\\SMC'
RAW = '\\Output\\xscratch\\RAW'
TAW = '\\Output\\xscratch\\TAW'
PET = '\\Output\\xscratch\\PET'
PERC = '\\Output\\xscratch\\Perc'

# ESRI ASCII grid header
ESRI_Header = {'ncols': 312,'nrows': 408,'xllcorner': 170291.27,'yllcorner':
231639.80,'cellsize': 400,'NODATA_value': -9999}
```

```
#==============================================================================
# RainINT
#
#    supporting Library for Rainfall Interception calculation
#
#    Python script by Changhui Park <Changhui.Park@umkc.edu>
#    Date 07.20.2011
#
# Hydro Lab.
# Department of Geosciences, University of Missouri - Kansas City
#==============================================================================

from numpy import array,exp

def cal_Interception(PPT,LAI):
    """
    Rainfall Interception calculation Function

    Input
        PPT: Precipitation [mm]
        LAI: Leaf Area Index
    Output
        rINT: Rainfall Interception [mm]
    """
    k = 0.065 * LAI # Aston (1979)
    SCmax = 0.935 + 0.498 * LAI - 0.00575 * LAI**2 #Von Hoyningen-Huene (1981)
    rINT = SCmax * (1 - exp(-k * PPT / SCmax)) # Marriam (1960) and Aston (1979)
    return rINT
```

```
#===============================================================================
# NRCSCNrunoff
#
#    supporting Library for NRCS CN method for runoff calculation
#
#    Python 2.5 code by Changhui Park <Changhui.Park@umkc.edu>
#    Date 07.20.2011
#
# Hydro Lab.
# Department of Geosciences, University of Missouri - Kansas City
#===============================================================================

from numpy import array,exp,log,where

def
cal_Runoff(PPT,CN2o,IaS_ratio=0.05,Slope=None,SMC=None,SMC_wp=None,SMC_fc=None,SMC_sa
t=None):
    """
    Direct Runoff Calculation by NRCS CN method

    Required Input
        PPT: Precipitation [mm]
        CN2o: Curve Number from the table
    Optional Input
        IaS_ratio: initial abstraction ratio (0.05 or 0.2)
        Slope: Surface Slope 0 to 1.0 (0% to 100%)
        SMC: Soil Moisture Content [mm]
        SMC_wp: Soil Moisture Content at wiliting point [mm]
        SMC_fc: Soil Moisture Content at field capacity [mm]
        SMC_sat: Soil Moisture Content at Saturation [mm]
    Output
        RO: Direct Runoff [mm]
    """
    CN1,CN2,CN3 = get_CNadj(CN2o,IaS_ratio,Slope)

    S = cal_S(CN1,CN2,CN3,SMC,SMC_wp,SMC_fc,SMC_sat)

    Ia = IaS_ratio * S
    RO = where(PPT <= Ia, 0.0, (PPT - Ia)**2.0 / (PPT - Ia + S))
    return RO

def get_CNadj(CN2o,IaS_ratio,Slope):
    """
    Slope and Ia/S (initial abstraction ratio) Adjustment

    Input
        CN2o: Curve Number from the table
        IaS_ratio: 0.05 or 0.2
        Slope: Surface Slope 0 to 1.0 (0% to 100%)
```

```
    Output
        CN1,CN2,CN3: Curve Numbers adjusted by slope and Ia/S
    """
    if Slope == None:
        CN2s = CN2o
    else:
        CN3o = (23.0 * CN2o) / (10.0 + 0.13 * CN2o)
        CN2s = ((CN3o - CN2o) / 3.0) * (1.0 - 2.0 * exp(-13.86 * Slope)) + CN2o #Slope
adjustment

    if IaS_ratio == 0.05:
        CN2 = 100.0 / (1.879 * (100.0 / CN2s - 1.0)**1.15 + 1.0) # CN0.2 to CN0.05
conversion
    else:
        CN2 = CN2s

    CN3 = (23.0 * CN2) / (10.0 + 0.13 * CN2)
    CN1 = (4.2 * CN2) / (10.0 - 0.058 * CN2)
    return CN1,CN2,CN3

def cal_S(CN1,CN2,CN3,SMC,SMC_wp,SMC_fc,SMC_sat):
    """
    Retention Parameter (Storage Index) Calculation

    Required Input
        CN2: Curve Number for ARC-II
    Optional Input
        CN1: Curve Number for ARC-I
        CN3: Curve Number for ARC-III
        SMC: Soil Moisture Content [mm]
        SMC_wp: Soil Moisture Content at wiliting point [mm]
        SMC_fc: Soil Moisture Content at field capacity [mm]
        SMC_sat: Soil Moisture Content at Saturation [mm]
    Output
        S = Retention Parameter
    """
    if SMC == None:
        S = 25.4 * (1000.0 / CN2 - 10.0)
    else:
        S = []
        for i in range(len(CN2)):
            if CN2[i] == 100.0:
                S.append(25.4 * (1000.0 / CN2[i] - 10.0))
            else:
                S1 = 25.4 * (1000.0 / CN1[i] - 10.0)
                S3 = 25.4 * (1000.0 / CN3[i] - 10.0)
                if SMC[i] <= SMC_wp[i]:
                    S.append(S1)
                elif SMC[i] == SMC_fc[i]:
```

```
            S.append(S3)
         elif SMC[i] == SMC_sat[i] or CN2[i] == 99:
            S.append(2.54)
         else:
            w2 = (log(SMC_fc[i] / (1 - S3 * S1**-1) - SMC_fc[i]) - log(SMC_sat[i]
/ (1 - 2.54 * S1**-1) - SMC_sat[i])) / (SMC_sat[i] - SMC_fc[i])
            w1 = log(SMC_fc[i] / (1 - S3 * S1**-1) - SMC_fc[i]) + w2 * SMC_fc[i]
            S.append(S1 * (1 - SMC[i] / (SMC[i] + exp(w1 - w2 * SMC[i]))))

   return array(S)
```

```
#===============================================================================
# EVTLIB
#
#   supporting Library for
#    FAO56 PM ETo (Allen et al., 1998) and Valiantzas's Eo and ETo (Valintzas, 2006)
#
#    Python 2.5 code by Changhui Park <Changhui.Park@umkc.edu>
#    Date 07.20.2011
#
# Hydro Lab.
# Department of Geosciences, University of Missouri - Kansas City
#===============================================================================

from math import pi
from numpy import array,sin,cos,tan,arccos,log,sqrt,exp,abs,maximum

def cal_EToPM(Tmean,Tmin,Tmax,RHmean,u2,EL,rLat,iDay,kRs=0.16):
   """
   Reference Evapotranspiration Function using FAO56 Equation 6

   Input
      Tmax: maximum daily temperature [C]
      Tmin: minimum daily temperature [C]
      Tmean: mean daily temperature [C]
      RHmean: mean daily relative humidity [%]
      u2: wind speed at 2 m height above ground surface [m]
      EL: station elevation above sea level [m]
      rLat: latitude in radian
      iDay: number of the day in the year between 1 to 365 or 366
      kRs: adjustment coefficient (0.16 to 0.19) interior to coastal area.
default=0.16
   Output
      ETo: Reference Evapotranpiration of the reference grass [mm/day]
   """
   DELTA = cal_DELTA(Tmean)
   gamma = cal_gamma(EL)
   Ra = cal_Ra(rLat,iDay); Rs = cal_Rs(kRs,Tmax,Tmin,Ra)
   Rso = cal_Rso(EL,Ra); Rns = cal_Rns(Rs)
   eoTmax = cal_eo(Tmax); eoTmin = cal_eo(Tmin)
   es = cal_es(eoTmax,eoTmin); ea = cal_ea(RHmean,eoTmax,eoTmin)
   Rn = cal_Rn(Tmax,Tmin,Rs,Rso,Rns,ea)

   # G (Soil Heat Flux) value is ignored by FAO Equation 42
   ETo = (0.408 * DELTA * Rn + gamma * 900.0 / (Tmean + 273.0) * u2 * (es - ea)) /
(DELTA + gamma * (1.0 + 0.34 * u2))
   return ETo

def cal_EVpen(Tmean,Tmin,Tmax,RHmean,u2,EL,rLat,iDay,kRs=0.16):
   """
```

```
    Penman Open water Evaporation Function by Penman (1948, 1963) in Valiantzas (2006)

    Input
        Tmax: maximum daily temperature [C]
        Tmin: minimum daily temperature [C]
        Tmean: mean daily temperature [C]
        RHmean: mean daily relative humidity [%]
        u2: wind speed at 2 m height above ground surface [m]
        EL: station elevation above sea level [m]
        rLat: latitude in radian
        iDay: number of the day in the year between 1 to 365 or 366
        kRs: adjustment coefficient (0.16 to 0.19) interior to coastal area.
default=0.16
    Output
        ETo: Reference Evapotranpiration of the reference grass [mm/day]
    """
    DELTA = cal_DELTA(Tmean)
    gamma = cal_gamma(EL)
    Ra = cal_Ra(rLat,iDay)
    Rs = cal_Rs(kRs,Tmax,Tmin,Ra)
    Rso = cal_Rso(EL,Ra)
    Rns = cal_Rns(Rs)
    eoTmax = cal_eo(Tmax)
    eoTmin = cal_eo(Tmin)
    es = cal_es(eoTmax,eoTmin)
    ea = cal_ea(RHmean,eoTmax,eoTmin)
    Rn = cal_Rn(Tmax,Tmin,Rs,Rso,Rns,ea)

    fu = 1.0 + 0.536 * u2 #wind function au + bu * u2 (au = 1, bu = 0.536)
    vpD = es - ea #vapor pressure deficit [kPa]
    Lambda = 2.257 #Latent heat of vaporization [MJ/kg] under air pressure

    part1 = DELTA / (DELTA + gamma)
    part2 = Rn / Lambda
    part3 = gamma / (DELTA + gamma)
    part4 = 6.43 * fu * vpD / Lambda

    EVpenman = part1 * part2 + part3 * part4

    return EVpenman

def cal_EToVa(Tmean,Tmin,Tmax,RHmean,u2,EL,rLat,iDay,kRs=0.16):
    """
    Reference Evapotranspiration Function using Valiantzas's Equation 36, 38, and 39
      Equation 36: Elevation adjustment
      Equation 38: Simplified ETo
      Equation 39: Simplified ETo when wind speed data is not available

    Input
```

```
      Tmax: maximum daily temperature [C]
      Tmin: minimum daily temperature [C]
      Tmean: mean daily temperature [C]
      RHmean: mean daily relative humidity [%]
      u2: wind speed at 2 m height above ground surface [m] (optional, if not
available, put 'None')
      EL: station elevation above sea level [m]
      rLat: latitude in radian
      iDay: number of the day in the year between 1 to 365 or 366
      kRs: adjustment coefficient (0.16 to 0.19) interior to coastal area.
default=0.16
   Output
      ETo: Reference Evapotranpiration of the reference grass [mm/day]
   """
   albedo = 0.25 # Value 0.25 shows better fit than value 0.23.
   Ra = cal_Ra(rLat,iDay)
   Rs = cal_Rs(kRs,Tmax,Tmin,Ra)

   if EL == None:
      EL = 0.0

   if u2 == None:
      ETo = 0.038 * Rs * sqrt(Tmean + 9.5) - 2.4 * (Rs / Ra)**2 + 0.075 \
          * (Tmean + 20.0) * (1 - RHmean / 100.0) + 0.00012 * EL
   else:
      ETo = 0.051 * (1.0 - albedo) * Rs * sqrt(Tmean + 9.5) - 2.4 * (Rs / Ra)**2 \
          + 0.048 * (Tmean + 20.0) * (1 - RHmean / 100.0) * (0.5 + 0.536 * u2) +
0.00012 * EL

   return ETo

def cal_EVopenVa(Tmean,Tmin,Tmax,RHmean,u2,EL,rLat,iDay,kRs=0.16):
   """
   Penman Open water Evaporation Function using Valiantzas's Equation 31, 33, and 36
      Equation 31: Simplified Penman Open water Evaporation
      Equation 33: Simplified penman Open water Evaporation when wind speed data is
not available
      Equation 36: Elevation adjustment

   Input
      Tmax: maximum daily temperature [C]
      Tmin: minimum daily temperature [C]
      Tmean: mean daily temperature [C]
      RHmean: mean daily relative humidity [%]
      u2: wind speed at 2 m height above ground surface [m] (optional, if not
available, put 'None')
      EL: station elevation above sea level [m]
      rLat: latitude in radian
      iDay: number of the day in the year between 1 to 365 or 366
```

```
      kRs: adjustment coefficient (0.16 to 0.19) interior to coastal area.
default=0.16
   Output
      Eopen: Reference Evapotranpiration of the reference grass [mm/day]
   """
   albedo = 0.08 # albedo for open water
   Ra = cal_Ra(rLat,iDay)
   Rs = cal_Rs(kRs,Tmax,Tmin,Ra)
   au = 0.5

   if EL == None:
      EL = 0.0

   if u2 == None:
      Eopen = 0.047 * Rs * sqrt(Tmean + 9.5) - 2.4 * (Rs / Ra)**2 + 0.09 * (Tmean +
20.0) * (1 - RHmean / 100.0) + 0.00012 * EL
   else:
      Eopen = 0.051 * (1.0 - albedo) * Rs * sqrt(Tmean + 9.5) - 0.188 * (Tmean +
13.0) * (Rs / Ra - 0.194) \
            * (1 - 0.00014 * (0.7 * Tmax + 0.3 * Tmin + 46.0)**2 * sqrt(RHmean /
100.0)) \
            + 0.049 * (Tmax + 16.3) * (1.0 - RHmean / 100.0) * (au + 0.536 * u2) +
0.00012 * EL

   return Eopen

def get_Kcadj(Kc_tab,Tmax,Tmin,u2,CrH):
   """
   Climate adjusted Kc based on FAO56 Equation 62

   Input
      Kc_tab: value for Kc taken from table 12 or 17
      Tmax, Tmin: Maximum and Minimum Temperature [C]
      u2: wind speed at 2 m above ground surface [m]
      crH: Mean Crop Height during the season [m] for 0.1 m ~ 10 m
   Output
      Kc_adj: Climate adjusted Kc
   """
   RHmin = cal_RHmin(Tmax,Tmin)
   Kc_adj = Kc_tab + (0.04 * (u2 -2.0) - 0.004 * (RHmin - 45.0)) * (CrH / 3.0)**0.3
   return Kc_adj

def get_Kcmax(Kcb_tab,Tmax,Tmin,u2,CrH,Kc_max0=1.2):
   """
   Upper limit on the EVT from any cropped surface using FAO56 Equation 72
   Returned values will be used for LAI calculation.

   Input
      Kcb_tab: Kcb taken from FAO56 Table 17
```

```
        Tmax, Tmin: Maximum and Minimum Temperature [C]
        u2: wind speed at 2 m above ground surface [m]
        CrH: Mean Crop Height during the season [m] for 0.1 m ~ 10 m
        Kc_max0: if more frequent wetting event (daily or each two days): 1.1
    Output
        Kc_max: Upper limit on the EVT
    """
    Kc_max1 = get_Kcadj(Kc_max0,Tmax,Tmin,u2,CrH)
    Kcb_adj = get_Kcadj(Kcb_tab,Tmax,Tmin,u2,CrH)
    Kc_max2 = Kcb_adj + 0.05
    Kc_max = maximum(Kc_max1,Kc_max2)
    return Kc_max

def cal_DELTA(Temp):
    """
    Slope of Saturation vapour pressure curve using FAO56 Equation 13

    Input
        Temp: Air Temperature [C]
    Output
        DELTA: Slope of Saturation vapour pressure at Temp [kPa/C]
    """
    DELTA = 4098.0 * (cal_eo(Temp)) / (Temp + 237.3)**2
    return DELTA

def cal_gamma(EL):
    """
    Psychrometric constant using FAO56 Equation 7 and 8

    Input
        EL: Elevation from sea level [m]
    Output
        gamma: Psychrometric constant at P [kPa/C]
    """
    P = 101.3 * ((293.0 - 0.0065 * EL) / 293.0)**5.26
    gamma = 0.665E-3 * P
    return gamma

def cal_Ra(rLat,iDay):
    """
    Extraterrestrial Radiation for daily periods using FAO56 Equation 21 and 23 to 25
        Equation 21: Extraterrestrial Radiation (Ra)
        Equation 23: Inverse relative distance Earth-Sun (dr)
        Equation 24: Solar declination (delta)
        Equation 25: Sunset hour angle (ws)

    Input
        rLat: latitude in radian
        iDay: the number of the day in the year between 1 and 365 (or 366)
```

```
    Output
        Ra: Extraterrestrial Radiation [MJ/m2/day]
    """
    Gsc = 0.082 #Solar constant [MJ m2/min]
    dr = 1.0 + 0.033 * cos(2.0 * pi / 365.0 * iDay) #Inverse relative distance Earth-
Sun
    delta = 0.409 * sin(2.0 * pi / 365.0 * iDay - 1.39) #Solar declination
    ws = arccos(-tan(rLat) * tan(delta)) #Sunset hour angle [radian]
    Ra = 1440.0 / pi * Gsc * dr * (ws * sin(rLat) * sin(delta) + cos(rLat) *
cos(delta) * sin(ws))
    return Ra #Extraterrestrial Radiation [MJ m2/day]

def cal_Rs(kRs,Tmax,Tmin,Ra):
    """
    Solar Radiation using FAO56 Equation 50

    Input
        kRs: adjustment coefficient (0.16 to 0.19) interior to coastal area
        Tmax: Maximum temperature during the day [C]
        Tmin: Minimum temperature during the day [C]
        Ra: Extraterrestrial Radiation [MJ/m2/day]
    Output
        Rs: Solar Radiation [MJ/m2/day]
    """
    Rs = kRs * sqrt(abs(Tmax - Tmin)) * Ra
    return Rs

def cal_Rso(EL,Ra):
    """
    Clear-sky Solar Radiation using FAO56 Equation 37

    Input
        EL: station elevation above sea level [m]
        Ra: extraterrestrial radiation [MJ/m2/day]
    Output
        Rso: Clear-sky Solar Radiation at EL [MJ/m2/day]
    """
    Rso = (0.75 + 2E-5 * EL) * Ra
    return Rso

def cal_Rns(Rs):
    """
    Net Shortwave Radiation using FAO56 Equation 38

    Input
        Rs: Solar Radiation [MJ/m2/day]
    Output
        Rns: Net Shortwave Radiation  [MJ/m2/day]
    """
```

```
    albedo = 0.23 #0.23: albedo for the reference grass
    Rns = (1 - albedo) * Rs
    return Rns

def cal_Rn(Tmax,Tmin,Rs,Rso,Rns,ea):
    """
    Net Radiation using FAO56 Equation 39 and 40
        Equation 39: Net Longwave radiation (Rnl)
        Equation 40: Net Radiation (Rn)

    Input
        Tmax: Maximum temperature during the day [C]
        Tmin: Minimum temperature during the day [C]
        ea: Actual Vapour Pressure [kPa]
        Rs: Solar Radiation [MJ/m2/day]
        Rso: Clear-sky Solar Radiation at EL [MJ/m2/day]
        Rns: Net Shortwave Radiation [MJ/m2/day]
    Output
        Rn: Net Radiation [MJ/m2/day]
    """
    sigma = 4.903E-9 #Stefan-Boltzmann Constant [MJ/K4/m2/day]
    TmaxK = Tmax + 273.16 #temperature conversion C to K
    TminK = Tmin + 273.16 #temperature conversion C to K
    Rnl = sigma * ((TmaxK**4 + TminK**4) / 2.0) * (0.34 - 0.14 * sqrt(ea)) * (1.35 *
Rs / Rso - 0.35) #Net Longwave Radiation [MJ/m2/day]
    Rn = Rns - Rnl
    return Rn

def cal_eo(Temp):
    """
    Saturation Vapour Pressure using FAO56 Equation 11

    Input
        Temp: Temperature [C]
    Output
        eo: Saturation Vapour Pressure at Temp [kPa]
    """
    eo = 0.6108 * exp(17.27 * Temp / (Temp + 237.3))
    return eo

def cal_es(eoTmax,eoTmin):
    """
    Saturation Vapour Pressure using FAO56 Equation 12

    Input
        eoTmax: Saturation Vapour Pressure at the maximum temperature [kPa]
        eoTmin: Saturation Vapour pressure at the minimum temperature [kPa]
    Output
        es: Mean Saturation Vapor Pressure [kPa]
```

```
    """
    es = (eoTmax + eoTmin) / 2.0
    return es

def cal_ea(RHmean,eoTmax,eoTmin):
    """
    Actual Vapour Pressure Function using FAO56 Equation 19

    Input
        RHmean: Mean Relative Humidity [%]
        eoTmax: Saturation Vapour Pressure at the maximum temperature [kPa]
        eoTmin: Saturation Vapour pressure at the minimum temperature [kPa]
    Output
        ea: Actual Vapour Pressure [kPa]
    """
    ea = RHmean / 100 * ((eoTmax + eoTmin) / 2.0)
    return ea

def cal_RHmin(Tmax,Tmin):
    """
    Minimum Relative Humidity Function using FAO56 Equation 64

    Input
        eoTmax: Saturation Vapour Pressure at the maximum temperature [kPa]
        eoTmin: Saturation Vapour pressure at the minimum temperature [kPa]
    Output
        RHmin: Minimum Relative Humidity [%]
    """
    eoTmin = cal_eo(Tmin)
    eoTmax = cal_eo(Tmax)
    RHmin = eoTmin / eoTmax * 100.0
    return RHmin

def con_u2(uz,z):
    """
    Wind speed conversion Function using FAO56 Equation 47

    Input
        uz: wind speed at height z above ground surface [m/s]
        z: height of measurement above ground surface [m]
    Output
        u2: wind speed at 2 m above ground surface for the reference grass [m/s]
    """
    u2 = uz * 4.87 / log(67.8 * z - 5.42)
    return u2
```

```
#===============================================================================
# ISWB (Improved Soil Water Balance Calculator)
#    Soil Moisture Deficit (SMD) based Soil Water Balance Calculator
#
#    Python 2.5 code by Changhui Park <Changhui.Park@umkc.edu>
#    Date: 01.17.2011
#
# Hydro Lab.
# Department of Geosciences, University of Missouri - Kansas City
#===============================================================================

from numpy import array,copy,zeros

def cal_gNSSWB(LCV,LcovTab,EVopen,sINF,PET,SURFSTORo,FRACSTOR,TAW,RAW,SMDo,GWS):
    '''
   Soil Water Balance on the ith day of the year (Grid)

   Inputs:
      sINF: infiltrated amount of water from surface on the ith day (PPT - INT -
ROF) [mm]
      PET: potential evapotranspiration on the ith day (Kc * ETo) [mm]
      SURFSTORo: Near Surface Soil water Storage from the previous day [mm]
      FRACSTOR: water holding capacity by soil type [-]
      TAW: Total Available Water [mm]
      RAW: Readily Available Water [mm]
      SMDo: Soil Moisture Deficit from the previous day

   Outputs:
      AET: Actual Evapotranspiration [mm]
      SURFSTOR: Near Surface Soil water Storage at the end of the ith day [mm]
      SMD: Soil Moisture Deficit at the end of the ith day [mm]
      PERC: Total amount of water existing the bottom of the soil profile on the ith
day [mm]
    '''
   AET = copy(PET)
   SURFSTOR = zeros(SURFSTORo.shape, 'float')
   SMD = zeros(SMDo.shape, 'float')
   PERC = zeros(SMDo.shape, 'float')

   for i in range(len(LCV)):
      if LcovTab[LCV[i]]['Land_Type'] == 'Water':
         AET[i] = EVopen[i]
         SURFSTOR[i] = 0.0
         PERC[i] = 0.0
         SMD[i] = 0.0
      else:
         # sINF correction for impervious cell
         if LcovTab[LCV[i]]['Land_Type'] == 'Impervious':
            if sINF[i] > EVopen[i]:
```

```
            sINF[i] = sINF[i] - EVopen[i]
         else:
            EVopen[i] = sINF[i]
            sINF[i] = 0.0

      INF = sINF[i] + SURFSTORo[i] + GWS[i]

      if INF > PET[i]:
         SURFSTOR[i] = FRACSTOR[i] * (INF - PET[i])
      else:
         SURFSTOR[i] = 0.0

      if SMDo[i] < RAW[i] or INF >= PET[i]:
         AET[i] = PET[i]
      elif (SMDo[i] <= TAW[i] and SMDo[i] >= RAW[i]) and INF < PET[i]:
         Kr = (TAW[i] - SMDo[i]) / (TAW[i] - RAW[i])
         AET[i] = INF + Kr * (PET[i] - INF)
      elif SMDo[i] >= TAW[i] and INF < PET[i]:
         AET[i] = INF

      SMD[i] = SMDo[i] - INF + SURFSTOR[i] + AET[i]
      if SMD[i] < 0:
         PERC[i] = -1 * SMD[i]
         SMD[i] = 0.0

      # AET correction for impervious cell
      if LcovTab[LCV[i]]['Land_Type'] == 'Impervious':
         AET[i] = AET[i] + EVopen[i]

return AET,SURFSTOR,SMD,PERC
```

```
#===============================================================================
# ActualRCH
#    Actual groundwater recharge calculator
#
#    Python 2.5 code by Changhui Park <Changhui.Park@umkc.edu>
#    Date: 01.17.2011
#
# Hydro Lab.
# Department of Geosciences, University of Missouri - Kansas City
#===============================================================================

from numpy import exp,array,argmax
from numpy.fft import rfft,irfft

def cal_aRCH(PERC,RCHo,deltaGW):
    """
    Actual groundwater recharge of the day

    Input
        PERC: the amount water from the bottom of the soil profile on i Day [mm]
        RCHo: the amount of acutal recharge on i-1 day [mm]
        deltaGW: time lag for water routing [day]
    Output
        aRCH: actual recharge of the day (i day) [mm]
    """
    aRCH = (1 - exp(-1 / deltaGW)) * PERC + exp(-1 / deltaGW) * RCHo
    return aRCH
```

```
#==============================================================================
# MODFLOW Driver for VELAS
#
#    Python 2.5 code by Changhui Park
#    Date: 7.15.2011
#
# Hydro Lab.
# Department of Geosciences, University of Missouri - Kansas City
#==============================================================================

from subprocess import call
from numpy import asarray,where,max,abs
import struct
import os


class MODFLOW:
    """
   MODFLOW class contains all methods and variables to launch the external
groundwater flow model.
    """
    def __init__(self, exeFPath, namFPath, numActiveCells):
        """
        initializer
        """
        self.MF2Kexe = exeFPath
        self.namFile = os.path.basename(namFPath)
        self.modPath = os.path.dirname(namFPath)
        self.numStress = 1
        self.openNAM(namFPath)

    def openNAM(self, namFPath):
        """
        This function is invoked by the initializer
        """
    # Open .nam file and get full paths for required files
        namF = file(namFPath)
        data = {}
        for line in namF:
            line = line.split()
            flag = line[0].upper()
            if flag == 'DIS':
                self.disFpath = os.path.join(self.modPath, line[2])
            elif flag == 'OC':
                self.ocFpath = os.path.join(self.modPath, line[2])
            elif flag == 'RCH':
                self.rchFpath = os.path.join(self.modPath, line[2])
            elif flag == 'BAS6':
                self.basFpath = os.path.join(self.modPath, line[2])
```

```
        elif flag == 'DATA':
            data[line[1]] = [os.path.join(self.modPath, line[2]), 'r']
        elif flag == 'DATA(BINARY)':
            data[line[1]] = [os.path.join(self.modPath, line[2]), 'rb']
        else:
            pass

# Open .dis file and get grid properties and time and length unit
    disF = file(self.disFpath)
    lineNum = 1
    for line in disF:
        line = line.upper().split()
        if '#' in line[0]:
            pass
        elif '#' not in line[0] and lineNum == 1:
            self.nrows = int(line[1])
            self.ncols = int(line[2])
            self.tUnit = int(line[4])
            self.lUnit = int(line[5])
            lineNum += 1
        elif '#' not in line[0] and lineNum == 2:
            lineNum += 1
        elif '#' not in line[0] and (lineNum == 3 or lineNum == 4):
            if line[0] == 'CONSTANT' or line[0] == 'OPEN/CLOSE':
                lineNum += 1
            else:
                disF.next()
                lineNum += 1
        elif '#' not in line[0] and lineNum == 5:
            if line[0].upper() == 'CONSTANT':
                self.topElev = float(line[1])
                break
            elif line[0] == 'OPEN/CLOSE':
                topElevFile = file(os.path.join(self.modPath, line[1]))
                elevData = topElevFile.read()
                self.topElev = asarray(map(float, elevData.split()))
                break
            else:
                topElev = []
                for i in range(self.nrows):
                    line = disF.next()
                    topElev += map(float, line.split())
                self.topElev = asarray(topElev)
                break
        else:
            pass

    self.topElev.shape = (self.nrows * self.ncols,)
```

```
   # Open .oc file and recognize head output location and type
      ocF = file(self.ocFpath)
      for line in ocF:
         line = line.upper()
         if 'HEAD SAVE UNIT' in line:
            line = line.split()
            self.curHedFpath, self.hedFmode = data[line[3]]
         elif 'HEAD SAVE FORMAT' in line:
            line = line.split()
            self.numValuesInaLine = int(line[3][1:3])
            if 'LABEL' in line:
               self.hedLabel = True
            else:
               self.hedLabel = False
         else:
            pass
      self.oldHedFpath = self.modPath + '\\old_' +
os.path.basename(self.curHedFpath)

   # Open .rch file and recognize its structure
      rchF = file(self.rchFpath)
      lineNum = 0
      rchFDs = []
      for line in rchF:
         lineNum += 1
         if 'INTERNAL' in line.upper():
            self.rchFnumHL = lineNum
            break
         elif 'OPEN/CLOSE' in line.upper():
            line = line.split()
            rchFDs.append(line[1])

      if rchFDs:
         self.rchFnumHL = 0
         if len(rchFDs) == 1:
            self.rchFpath = os.path.join(self.modPath, rchFDs[0])
         else:
            self.rchF1path = os.path.join(self.modPath, rchFDs[0])
            self.rchF2path = os.path.join(self.modPath, rchFDs[1])

   # Open .ba6 file and get boundary condition as array.
      basF = file(self.basFpath)
      for line in basF:
         bndArray = []
         if 'INTERNAL' in line.upper():
            for i in range(self.nrows):
               line = basF.next()
               bndArray += map(int, line.split())
            self.bndArray = asarray(bndArray)
```

```
            break

        elif 'OPEN/CLOSE' in line.upper():
            line = line.split()
            bndF = file(os.path.join(self.modPath, line[1]))
            bndData = bndF.read()
            self.bndArray = asarray(map(int, bndData.split()))
            break

        else:
            pass

    self.bndArray.shape = (self.nrows * self.ncols,)

def run(self, rchGrids):
    """
    Call MODFLOW executable with RCH grid(s)
    """

    self.rch2MOD(rchGrids)

    if os.path.exists(self.oldHedFpath):
        os.remove(self.oldHedFpath)
    if os.path.exists(self.curHedFpath):
        os.rename(self.curHedFpath, self.oldHedFpath)

    curDir = os.getcwd()
    os.chdir(self.modPath)
    call([self.MF2Kexe, self.namFile])
    os.chdir(curDir)

    curHeds = self.hedReader(self.curHedFpath, self.hedFmode)
    if os.path.exists(self.oldHedFpath):
        oldHeds = self.hedReader(self.oldHedFpath, self.hedFmode)
        return self.hed2gwdGrid(curHeds)
    else:
        return self.hed2gwdGrid(curHeds)


def rch2MOD(self,rchGrids):
    """
    Converter for Recharge output to MODFLOW rch input.
    rchGrids should be List that is containing Winter and Summer recharge output
or Year output as array type.
    """
    # MODFLOW rch input file preparation.
    rchF = [] # file objects container
    oldRch = self.modPath+'\\old_'+os.path.basename(self.rchFpath)
        if os.path.exists(oldRch): os.remove(oldRch)
```

```
            if os.path.exists(self.rchFpath): os.rename(self.rchFpath, oldRch)
            rchF.append(file(self.rchFpath, 'w'))

        # mm/yr to m/day conversion
        for i in range(len(rchGrids)):
            rchGrids[i] = where(self.bndArray != 0, rchGrids[i] * 0.001 / 365.0,
rchGrids[i])
            rchGrids[i].shape = (self.nrows, self.ncols)

        # rch data in the same file or seperated in two files
        if self.rchFnumHL > 0:
            headers = []
            oldRchF = file(oldRch)
            for line in range(self.rchFnumHL):
                headers.append(oldRchF.readline())
            oldRchF.close()
            rchF[0].writelines(headers[:-2])
            for rch in rchGrids:
                rchF[0].writelines(headers[-2:])
                for i in range(self.nrows):
                    strRow = ' '.join(map(str, rch[i]))
                    rchF[0].write(strRow + '\n')
        else:
            for i in range(len(rchF)):
                for j in range(self.nrows):
                    strRow = ' '.join(map(str, rchGrids[i][j]))
                    rchF[i].write(strRow + '\n')

        for i in range(len(rchF)):
            rchF[i].close()

    def hed2gwdGrid(self, curHeds):
        """
        This function should be invoked only in case of two stress periods
        """
        gwd0 = where(self.bndArray != 0, self.topElev - curHeds[0], -9999) # -9999 is
NoData in the ESRI raster form

        return [gwd0]

    def hedReader(self, hedFpath, hedFmode):
        hedF = file(hedFpath, hedFmode)
        hedValue = [[],[]]
        if hedFmode == 'r':
            for hed in hedValue:
                if self.hedLabel:
                    hedF.readline()
                if self.numValuesInaLine <= self.ncols:
                    for line in range(self.nrows):
```

```
            line = hedF.readline()
            hed.append(map(float, line.split()))
      else:
         Quo, Rem = divmod(self.ncols, self.numValuesInaLine)
         if Rem:
            numLines = (Quo + 1) * self.nrows
         else:
            numLines = Quo * self.nrows
         for line in range(numLines):
            line = hedF.readline()
            hed.append(map(float, line.split()))
else:
   for hed in hedValue:
      hedF.read(44)
      for value in range(self.nrows * self.ncols):
         value = struct.unpack('f', hedF.read(4))[0]
         hed.append(value)

hedValue[0] = asarray(hedValue[0])
hedValue[0].shape = (self.nrows * self.ncols,)
hedValue[1] = asarray(hedValue[1])
hedValue[1].shape = (self.nrows * self.ncols,)

return hedValue
```

REFERENCES

Abbott, M. B., J. C. Bathurst, J. A. Cunge, P. E. O'Connell, and J. Rasmussen. 1986. An introduction to the european hydrological system — Système Hydrologique Européen, “SHE”, 1: History and philosophy of a physically-based, distributed modelling system. *Journal of Hydrology* 87, no. 1-2: 45-59.

_____. 1986. An introduction to the european hydrological system — Système Hydrologique Européen, “SHE”, 2: Structure of a physically-based, distributed modelling system. *Journal of Hydrology* 87, no. 1-2: 61-77.

Allen, Richard G., Luis S. Pereira, Dirk Raes, and Martin Smith. 1998. *Crop evapotranspiration: Guidelines for computing crop water requirements*. FAO irrigation and drainage paper. Rome, Italy: FAO - Food and Agriculture Organization of the United Nation.

Allen, Richard G., William O. Pruitt, Dirk Raes, Martin Smith, and Luis S. Pereira. 2005. Estimating evaporation from bare soil and the crop coefficient for the initial period using common soils information. *Journal of Irrigation and Drainage Engineering* 131, no. 1: 14-23.

Alvisi, Stefano and Marco Franchini. 2011. Fuzzy neural networks for water level and discharge forecasting with uncertainty. *Environmental Modelling & Software* 26, no. 4: 523-537.

Arnold, Jeffrey G., Peter M. Allen, and Gilbert Bernhardt. 1993. A comprehensive surface-groundwater flow model. *Journal of Hydrology* 142, no. 1–4: 47-69.

Asner, Gregory P., Jonathan M. O. Scurlock, and Jeffrey A. Hicke. 2003. Global synthesis of leaf area index observations: Implications for ecological and remote sensing studies. *Global Ecology and Biogeography* 12, no. 3: 191-205.

Aston, A.R. 1979. Rainfall interception by eight small trees. *Journal of Hydrology* 42, no. 3–4: 383-396.

Aydin, Mehmet, Sheng Li Yang, Nurten Kurt, and Tomohisa Yano. 2005. Test of a simple model for estimating evaporation from bare soils in different environments. *Ecological Modelling* 182, no. 1: 91-105.

Baltas, E. A., N. A. Dervos, and M. A. Mimikou. 2007. Technical note: Determination of the SCS initial abstraction ratio in an experimental watershed in Greece. *Hydrology and Earth System Sciences* 11, no. 6: 1825-1829.

Basheer, I. A. and M. Hajmeer. 2000. Artificial neural networks: Fundamentals, computing, design, and application. *Journal of Microbiological Methods* 43, no. 1: 3.

Batelaan, O., Zhong-Min Wang, and F. De Smedt. 1996. An adaptive GIS toolbox for hydrological modelling. In *HydroGIS 96: Application of Geographic Information Systems in Hydrology and Water Resources Management*, ed. K. Kovar and H. P. Nachtnabel. Vienna: IAHS.

Batelaan, O. and F. De Smedt. 2001. Wetspass: A flexibel, GIS based, distributed recharge methodology for regional groundwater modeling. In *Impact of Human Activitiy on Groundwater Dynamics*, ed. H. Gehrels, J. Peters, E. Hoehn, K. Jensen, C. Leibundgut, J. Griffioen, B. Webb and W.-J. Zaadnoordijk:11-17. Wallingford: vol. Publ. No. 269. IAHS.

Batelaan, Okke. 2006. Characterizing groundwater recharge and discharge using remote sensing, GIS, ecology, hydrochemistry and groundwater modelling. Doctor of Philosophy, Vrije Universiteit Brussel.

Batelaan, Okke and F. De Smedt. 2007. GIS-based recharge estimation by coupling surface-subsurface water balances. *Journal of Hydrology* 337, no. 3-4: 337-355.

Batelaan, O. and S. T. Woldeamlak. 2007. ArcView interface for WetSpass - user manual 13-06-2007. Department of Hydrology and Hydraulic Engineering, Vrije Universiteit Brussel.

Bonsu, Mensah. 1997. Soil water management implications during the constant rate and the falling rate stages of soil evaporation. *Agricultural Water Management* 33, no. 2–3: 87-97.

Brisson, Nadine, Bruno Mary, Dominique Ripoche, Marie Hélène Jeuffroy, Françoise Ruget, Bernard Nicoullaud, Philippe Gate, Florence Devienne-Barret, Rodrigo Antonioletti, Carolyne Durr, Guy Richard, Nicolas Beaudoin, Sylvie Recous, Xavier Tayot, Daniel Plenet, Pierre Cellier, Jean-Marie Machet, Jean Marc Meynard, and Richard Delécolle. 1998. Stics: A

generic model for the simulation of crops and their water and nitrogen balances. I. Theory and parameterization applied to wheat and corn. *Agronomie* 18, no. 5-6: 311-346.

Brooks, Kenneth N., Peter F. Ffolliott, Hans M. Gregersen, and Leonard F. DeBano. 2003. *Hydrology and the management of watersheds*. Ames, Iowa: Iowa State Press.

Brutsaert, Wilfried. 2005. *Hydrology : An introduction*. New York, NY: Cambridge University Press.

Castillo, V. M., A. Gómez-Plaza, and M. Martínez-Mena. 2003. The role of antecedent soil water content in the runoff response of semiarid catchments: A simulation approach. *Journal of Hydrology* 284, no. 1–4: 114-130.

Chen, Jing M., Xiaoyong Chen, Weimin Ju, and Xiaoyuan Geng. 2005. Distributed hydrological model for mapping evapotranspiration using remote sensing inputs. *Journal of Hydrology* 305, no. 1-4: 15-39.

Chung, Il-Moon, Nam-Won Kim, Jeongwoo Lee, and Marios Sophocleous. 2010. Assessing distributed groundwater recharge rate using integrated surface water-groundwater modelling: Application to Mihocheon watershed, south Korea. *Hydrogeology Journal* 18, no. 5: 1253-1264.

Clark, C. 2002. Measured and estimated evaporation and soil moisture deficit for growers and the water industry. *Meteorological Applications* 9, no. 01: 85-93.

Clark Labs. 2009. IDRISI Taiga 16.05. Clark Labs, Worcester, MA.

Coe, Michael T., Marcos H. Costa, and Britaldo S. Soares-Filho. 2009. The influence of historical and potential future deforestation on the stream flow of the Amazon River – Land surface processes and atmospheric feedbacks. *Journal of Hydrology* 369, no. 1-2: 165-174.

Cohen, S., A. Ianetz, and G. Stanhill. 2002. Evaporative climate changes at bet dagan, israel, 1964–1998. *Agricultural and Forest Meteorology* 111, no. 2: 83-91.

Coleman, A. M. 2008. *An adaptive landscape classification procedure using geoinformatics and artificial neural networks*. PNNL-17777. Richland, WA: Pacific Northwest National Laboratory.

Cruise, James F., Charles A. Laymon, and Osama Z. Al-Hamdan. 2010. Impact of 20 years of land-cover change on the hydrology of streams in the southeastern United States. *JAWRA Journal of the American Water Resources Association* 46, no. 6: 1159-1170.

Dams, J., S. T. Woldeamlak, and O. Batelaan. 2008. Predicting land-use change and its impact on the groundwater system of the Klein Nete catchment, Belgium. *Hydrology and Earth System Sciences* 12, no. 6: 1369-1385.

Davis, John C. 2002. Statistics and data analysis in geology. New York: John Wiley & Sons.

De Jong, S. M. and V. G. Jetten. 2007. Estimating spatial patterns of rainfall interception from remotely sensed vegetation indices and spectral

mixture analysis. *International Journal of Geographical Information Science* 21, no. 5: 529-545.

de Silva, C. Shanthi and K. R. Rushton. 2008. Representation of rainfed valley ricefields using a soil–water balance model. *Agricultural Water Management* 95, no. 3: 271-282.

Domenico, P. A. and F. W. Schwartz. 1990. *Physical and chemical hydrogeology*. New York, NY: Wiley.

Eastman, Ron. 2009. *IDRISI Taiga: Guide to GIS and Image Processing*. Worcester, MA.

Eckhardt, K. 2005. How to construct recursive digital filters for baseflow separation. Hydrological Processes 19, no. 2: 507-515.

Eilers, V. H. M., R. C. Carter, and K. R. Rushton. 2007. A single layer soil water balance model for estimating deep drainage (potential recharge): An application to cropped land in semi-arid north-east Nigeria. *Geoderma* 140, no. 1-2: 119-131.

ESRI. 2008.ArcGIS desktop 9.3. Environmental Systems Research Institute, Redlands, CA.

Fetter, C. W. 2001. *Applied hydrogeology*. Upper Saddle River, NJ: Prentice Hall.

Graham, D. N. and M. B. Butts. 2005. Flexible, integrated watershed modeling with MIKE SHE. In *Watershed Models*, ed. V.P. Singhnad Donald K. Frevert: 245-272. Boca Raton, FL: CRC Press.

Grell, Georg A., Jimy Dudhia, and David R. Stauffer. 1995. *A description of the fifth-*

generation penn state/ncar mesoscale model (mm5). NCAR/TN-398 + STR.

Guo, Shenglian. 2002. Two-parameter monthly water balance model. In *Mathematical models of small watershed hydrology and applications*, ed. V. P. Singh and Donald K. Frevert:113-166. Highlands Ranch, CO: Water Resources Publications.

Gupta, Sushil K. 2010. *Modern hydrology and sustainable water development*. Hoboken, NJ: John Wiley & Sons.

Harbaugh, Arlen W., Edward R. Banta, Mary C. Hill, and Michael G. McDonald. 2000. *MODFLOW-2000, the U.S. Geological Survey modular ground-water model - user guide to modularization concepts and the ground-water flow process*. U.S. Geological Survey.

Hawkins, Richard H., Timothy J. Ward, Donald E. Woodward, and Joseph A. Van Mullem. 2010. Continuing evolution of rainfall-runoff and the curve number precedent. Paper presented at 2nd Joint Federal Interagency Conference, Las Vegas, NV, June 27-July 1.

Henriksen, Hans Jørgen, Lars Troldborg, Per Nyegaard, Torben Obel Sonnenborg, Jens Christian Refsgaard, and Bjarne Madsen. 2003. Methodology for construction, calibration and validation of a national hydrological model for Denmark. *Journal of Hydrology* 280, no. 1-4: 52-71.

Huang, Zhihua, Bin Xue, and Yong Pang. 2009. Simulation on stream flow and nutrient loadings in Gucheng Lake, Low Yangtze River Basin, based on SWAT model. *Quaternary International* 208, no. 1–2: 109-115.

Hwang, Jae Ha, Byung Joo Lee, and Kyo Young Song. 1994. Quaternary tectonic movement on cheju island. *Economic and Environmental Geology* 27, no. 2: 209-212.

Im, Sangjun, Seungwoo Park, and Taeil Jang. 2007. Application of SCS curve number method for irrigated paddy field. *KSCE Journal of Civil Engineering* 11, no. 1: 51-56.

IPCC. 2000. *Special report on emission scenarios*. Ed Nebojsa Nakicenovic and Rob Swart. Cambridge: Cambridge University Press.

IPCC. 2007. *Climate change 2007: Synthesis report*. Ed Core Writing Team, Rajendra K. Pachauri and Andy Reisinger. Contribution of working groups i, ii and iii to the fourth assessment report of the intergovernmental panel on climate change Geneva: IPCC.

Jarvis, A., H. I. Reuter, A. Nelson, and E. Guevara. 2008. Hole-filled seamless SRTM data V4. International Centre for Tropical Agriculture (CIAT).

Kang, M. S., S. W. Park, J. J. Lee, and K. H. Yoo. 2006. Applying swat for TMDL programs to a small watershed containing rice paddy fields. *Agricultural Water Management* 79, no. 1: 72-92.

Kannan, Narayanan, Chinnasamy Santhi, and Jeffrey G. Arnold. 2008. Development of an automated procedure for estimation of the spatial variation of runoff in large river basins. *Journal of Hydrology* 359, no. 1–2: 1-15.

Kim, Nam Won and Jeongwoo Lee. 2008. Temporally weighted average curve

number method for daily runoff simulation. *Hydrological Processes* 22, no. 25: 4936-4948.

Kim, Nam Won, Il Moon Chung, Yoo Seung Won, and Jeffrey G. Arnold. 2008. Development and application of the integrated SWAT–MODFLOW model. *Journal of Hydrology* 356, no. 1–2: 1-16.

Kim, Sun-Hwa, Ji-Hoon Park, Choong-Sik Woo, and Kyu-Sung Lee. 2005. Analysis of temporal variability of modis leaf area index (LAI) product over temperate forest in korea. In *Geosciences and Remote Sensing Symposium, 2005. IGARSS '05.*, 6:4343-4346: IEEE International.

King, K. W., J. G. Arnold, and R. L. Bingner. 1999. Comparison of Green-Ampt and curve number methods on Goodwin Creek watershed using SWAT. *Transactions of the ASAE* 42, no. 4: 919-925.

Kite, Geoff. 2002. Modeling the Olifants Basin.84: International Water Management Institute.

KOWACO. 2003. *Comprehensive investigation of hydrogeology and groundwater resources on jeju island*. Jeju Special Self-Governing Province, Korea Water Resources Coporation (KOWACO).

Kozak, Joseph A., Lajpat R. Ahuja, Timothy R. Green, and Liwang Ma. 2007. Modelling crop canopy and residue rainfall interception effects on soil hydrological components for semi-arid agriculture. *Hydrological Processes* 21, no. 2: 229-241.

Legutke, Stephanie and Reinhard Voss. 1999. *The hamburg atmosphere-ocean*

coupled circulation model ECHO-G. Hamburg: DKRZ (German Climate Computer Center).

Li, Haibin, Alan Robock, and Martin Wild. 2007. Evaluation of intergovernmental panel on climate change fourth assessment soil moisture simulations for the second half of the twentieth century. *Journal of Geophysical Research* 112, no. D6: D06106.

Li, Qiongfang and John Gowing. 2005. A daily water balance modelling approach for simulating performance of tank-based irrigation systems. *Water Resources Management* 19, no. 3: 211-231.

Lin, Yong and Xiaohua Wei. 2008. The impact of large-scale forest harvesting on hydrology in the Willow watershed of Central British Columbia. *Journal of Hydrology* 359, no. 1-2: 141-149.

Linacre, Edward T. 1993. Data-sparse estimation of lake evaporation, using a simplified penman equation. *Agricultural and Forest Meteorology* 64, no. 3–4: 237-256.

Loaiciga, Hugo A., Juan B. Valdes, Richard Vogel, Jeff Garvey, and Harry Schwarz. 1996. Global warming and the hydrologic cycle. *Journal of Hydrology* 174, no. 1–2: 83-127.

Merriam, Robert A. 1960. A note on the interception loss equation. *Journal of Geophysical Research* 65, no. 11: 3850-3851.

Monteith, J.L. 1965. Evaporation and environment. In *In: Proceedings of the 19th Symposium of the Society for Experimental Biology*:205-233. New York, NY:

Cambridge University Press.

Moretti, G. and A. Montanari. 2007. AFFDEF: A spatially distributed grid based rainfall–runoff model for continuous time simulations of river discharge. *Environmental Modelling & Software* 22, no. 6: 823-836.

Muzylo, A., P. Llorens, F. Valente, J. J. Keizer, F. Domingo, and J. H. C. Gash. 2009. A review of rainfall interception modelling. *Journal of Hydrology* 370, no. 1–4: 191-206.

Nathan, R. J. and T. A. McMahon. 1990. Evaluation of automated techniques for base flow and recession analyses. *Water Resource Research* 26, no. 7: 1465-1473.

Neitsch, S.L., J.G. Arnold, J.R. Kiniry, and J.R. Williams. 2005. Soil and water assessment tool theoretical documentation version 2005. Temple, TX: USDA.ARS Grassland, Soil and Water Research Laboratory.

NIMR. 2004. *The development of regional climate change scenario for the national climate change report (iii)*. Seoul: NIMR.

Oeurng, Chantha, Sabine Sauvage, and José-Miguel Sánchez-Pérez. 2011. Assessment of hydrology, sediment and particulate organic carbon yield in a large agricultural catchment using the SWAT model, *Journal of Hydrology* 401, no. 3–4: 145-153.

Oñate-Valdivieso, F. and Joaquín Bosque Sendra. 2010. Application of gis and remote sensing techniques in generation of land use scenarios for hydrological modeling. *Journal of Hydrology* 395, no. 3-4: 256-263.

Penman, H. L. 1948. Natural evaporation from open water, bare soil and grass. Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences 193, no. 1032: 120-145.

Penman, H. L. 1956. Evaporation: An introductory survey. *Netherlands Journal of Agricultural Science* 4: 9-29.

Penna, D., H. J. Tromp-van Meerveld, A. Gobbi, M. Borga, and G. Dalla Fontana. 2011. The influence of soil moisture on threshold runoff generation processes in an alpine headwater catchment. *Hydrology and Earth System Sciences* 15, no. 3: 689-702.

Pijanowski, Bryan C., Daniel G. Brown, Bradley A. Shellito, and Gaurav A. Manik. 2002. Using neural networks and GIS to forecast land use changes: A land transformation model. *Computers, Environment and Urban Systems* 26, no. 6: 553-575.

Pisinaras, Vassilios, Christos Petalas, Georgios D. Gikas, Alexandra Gemitzi, and Vassilios A. Tsihrintzis. 2010. Hydrological and water quality modeling in a medium-sized basin using the Soil and Water Assessment Tool (SWAT). *Desalination* 250, no. 1: 274-286.

Ponce, V. M. and R. H. Hawkins. 1996. Runoff curve number: Has it reached maturity? *Journal of Hydrologic Engineering* 1, no. 1: 11-18.

Pontius Jr, R. Gil. 2000. Quantification error versus location error in comparison of categorical maps. *Photogrammetric Engineering & Remote Sensing* 66, no. 8: 1011-1016.

Pontius Jr, R. Gil and Laura C. Schneider. 2001. Land-cover change model validation by an ROC method for the Ipswich watershed, Massachusetts, USA. *Agriculture, Ecosystems & Environment* 85, no. 1–3: 239-248.

Ragab, Ragab and John Bromley. 2010. IHMS-integrated hydrological modelling system. Part 1. Hydrological processes and general structure. *Hydrological Processes* 24, no. 19: 2663-2680.

Ramírez, Jorge A. and Sharika U. S. Senarath. 2000. A statistical–dynamical parameterization of interception and land surface–atmosphere interactions. *Journal of Climate* 13, no. 22: 4050-4063.

Rawls, W.J., L.R. Ahuja, D.L. Brakensiek, and A. Shirmohammadi. 1992. Infiltration and soil water movement. In *Handbook of hydrology*, ed. D.R. Maidment:5.1-5.5. New York, NY: McGraw-Hill, Inc.

Ritchie, Joe T. 1972. Model for predicting evaporation from a row crop with incomplete cover. Water Resource Research 8, no. 5: 1204-1213.

Rushton, K. R. 2003. *Groundwater hydrology : Conceptual and computational models*. Hoboken, NJ: Wiley.

Rushton, K. R., V. H. M. Eilers, and R. C. Carter. 2006. Improved soil moisture balance methodology for recharge estimation. *Journal of Hydrology* 318, no. 1-4: 379-399.

Sato, Tomonori, Fujio Kimura, and Akio Kitoh. 2007. Projection of global warming onto regional precipitation over Mongolia using a regional climate model. *Journal of Hydrology* 333, no. 1: 144-154.

Saxton, K. E. and W. J. Rawls. 2006. Soil water characteristic estimates by texture and organic matter for hydrologic solutions. *Soil Science Society of America Journal* 70, no. 5: 1569-1578.

Schneider, Laura C. and R. Gil Pontius Jr. 2001. Modeling land-use change in the Ipswich watershed, Massachusetts, USA. *Agriculture, Ecosystems & Environment* 85, no. 1–3: 83-94.

Schulze, R. E. 1995. Hydrology and agrohydrology: A text to accompany the ACRU 3.00 agrohydrological modelling system. Pretoria, Republic of South Africa: Water Research Commission.

Scipal, K., C. Scheffler, and W. Wagner. 2005. Soil moisture-runoff relation at the catchment scale as observed with coarse resolution microwave remote sensing. *Hydrology and Earth System Sciences* 9, no. 3: 173-183.

Sheikh, Vahedberdi, Saskia Visser, and Leo Stroosnijder. 2009. A simple model to predict soil moisture: Bridging event and continuous hydrological (BEACH) modelling. *Environmental Modelling & Software* 24, no. 4: 542-556.

Smerdon, B. D., D. M. Allen, and D. Neilsen. 2010. Evaluating the use of a gridded climate surface for modelling groundwater recharge in a semi-arid region (Okanagan Basin, Canada). *Hydrological Processes* 24, no. 21: 3087-3100.

Sophocleous, Marios. 2004. Groundwater recharge. In *Groundwater*, ed. Luis Silveira and Eduardo J. Usunoff, 1. Oxford, UK: EOLSS Publishers.

Sophocleous, Marios and Samuel P. Perkins. 2000. Methodology and application

of combined watershed and ground-water models in Kansas. *Journal of Hydrology* 236, no. 3–4: 185-201.

Suriya, S. and B.V. Mudgal. 2012. Impact of urbanization on flooding: The Thirusoolam sub watershed – A case study. *Journal of Hydrology* 412-413: 210-219.

Tarboton, David G. Rainfall-runoff processes. http://hydrology.usu.edu/RRP/ (accessed September 23, 2010).

Tilahun, Ketema and Broder J. Merkel. 2009. Estimation of groundwater recharge using a GIS-based distributed water balance model in Dire Dawa, Ethiopia. *Hydrogeology Journal* 17, no. 6: 1443-1457.

USDA-NRCS. 2004. Chapter 10: Estimation of direct runoff from storm rainfall. In *National engineering handbook, part 630 hydrology*. Washington, DC: United States Department of Agriculture - Natural Resources Conservation Service.

USDA-NRCS Conservation Engineering Division. 1986. *Urban hydrology for small watersheds*. Washington, DC: United States Department of Agriculture - Natural Resources Conservation Services.

USDA-NRCS Soil Survey Division. 1993. *Soil survey manual*. United states department of agriculture handbook. Washington, D.C.: United States Department of Agriculture - Natural Resources Conservation Services.

Valiantzas, John D. 2006. Simplified versions for the Penman evaporation equation using routine weather data. *Journal of Hydrology* 331, no. 3-4:

690.

van Dijk, A. I. J. M. and L. A. Bruijnzeel. 2001. Modelling rainfall interception by vegetation of variable density using an adapted analytical model. Part 1. Model description. *Journal of Hydrology* 247, no. 3–4: 230-238.

von Hoyningen-Huene, J. 1983. Die interzeption des niederschlages in landwirtschaftlichen pflanzenbeständen. In *Dvwk schriften 57 - einfluss der landnutzung auf den gebietswasserhaushalt*, ed. Deutscher Verband für Wasserwirtschaft und Kulturbau, 57:1-53. Hambrug, Germany: Paul Parey.

Ward, Andrew D. and Stanley Wayne Trimble. 2004. *Environmental hydrology*. Boca Raton, FL: Lewis Publishers.

White, Michael D. and Keith A. Greer. 2006. The effect of watershed urbanization on the stream hydrology and riparian vegetation of Los Peñasquitos Creek, California. *Landscape and Urban planning* 74, no. 2: 125-138.

Wilcox, Bradford P., W. J. Rawls, D. L. Brakensiek, and J. Ross Wight. 1990. Predicting runoff from rangeland catchments: A comparison of two models. *Water Resource Research* 26, no. 10: 2401-2410.

Williams, J. R., R. C. Izaurralde, and E. M. Steglich. 2008. *Agricultural Poilcy/Environmental eXtender model: Theoretical documentation version 0604*. Temple, TX: Texas AgriLIFE Research, Texas A&M University, Blackland Research and Extension Center.

Winter, Thomas C. 1998. *Ground water and surface water : A single resource*. U.S. Geological Survey circular. Denver, CO: U.S. Geological Survey.

Woodward, Donald E., Richard H. Hawkins, Ruiyun Jiang, Jr Allen T. Hjelmfelt, Joseph A. Van Mullem, and Quan D. Quan. 2003. Runoff curve number method: Examination of the initial abstraction ratio. ed. Paul Bizier and Paul DeBarry, 118:308-308. Philadelphia, Pennsylvania, USA: ASCE.

Wu, Kangsheng and Carol A. Johnston. 2007. Hydrologic response to climatic variability in a Great Lakes Watershed: A case study with the SWAT model. *Journal of Hydrology* 337, no. 1–2: 187-199.

Xie, Xianhong and Yuanlai Cui. 2011. Development and test of SWAT for modeling hydrological processes in irrigation districts with paddy rice, *Journal of Hydrology* 396, no. 1–2: 61-71.

Zhang, Y. K. and K. E. Schilling. 2006. Effects of land cover on water table, soil moisture, evapotranspiration, and groundwater recharge: A field observation and analysis. *Journal of Hydrology* 319, no. 1-4: 328-338.

Zhang, Zhiqiang, Shengpin Wang, Ge Sun, Steven G. McNulty, Huayong Zhang, Jianlao Li, Manliang Zhang, Eduard Klaghofer, and Peter Strauss. 2008. Evaluation of the MIKE SHE Model for Application in the Loess Plateau, China. *JAWRA Journal of the American Water Resources Association* 44, no. 5: 1108-1120.

VITA

Changhui Park was born in Cheonan, Korea on July 15, 1974. He attended Kongju National University in Korea and received his Bachelor of Science degree in Geoenvironmental Sciences in 2000. He was awarded the Master of Science degree in Geological Scineces in 2002 from Kongju National University. After the completion of master's program, he worked for the Nondestructive Research Laboratory of Cultural Property in Kongju National University, Korea for three years, 2002-2004, as a researcher. He began work toward his I.Ph.D. in Geosciences and Computer Science at the University of Missouri – Kansas City in the Fall of 2004. He worked as a Graduate Teaching Assistant in the Department of Geosciences.

During his research at the University of Missouri – Kansas City, he earned the Advanced Certificate in Geographic Information System. He participated in the Lake Chad project funded by NASA and NSF as a research assistant. His duty was constructing water balance model and modeling groundwater flow.

Mr. Changhui Park is a member of the American Geophysical Union.