UNIVERSITY OF MINNESOTA
ST. ANTHONY FALLS LABORATORY
Engineering, Environmental and Geophysical Fluid Dynamics

Project Report No. 408

Modifications of the Soil and Water Assessment Tool (SWAT) for Application to Climate Change Studies

by

Michael P. Hanratty and Heinz G. Stefan

Prepared for

U. S. DEPARTMENT OF AGRICULTURE
Agricultural Research Service
Grazingland Research Laboratory
El Reno, Oklahoma

November 1997
Minneapolis, Minnesota

UNIVERSITY OF MINNESOTA
ST. ANTHONY FALLS LABORATORY
Engineering, Environmental and Geophysical Fluid Dynamics

Project Report No. 408

Modifications of the Soil and Water Assessment Tool (SWAT) for Application to Climate Change Studies

by

Michael P. Hanratty and Heinz G. Stefan

Prepared for

U. S. DEPARTMENT OF AGRICULTURE
Agricultural Research Service
Grazingland Research Laboratory
El Reno, Oklahoma

November 1997
Minneapolis, Minnesota

Prepared for: USDA, ARS
Last Revised: 11-4-97
Disk Locators: (Zip Disk #2/c:Winword\docs\
PR408cov.doc; R408TXT1-3.doc)

ABSTRACT

The Soil and Water Assessment Tool (SWAT) is a water quality model developed by the Agricultural Research Service (ARS) to simulate runoff from agricultural watersheds. In order to obtain the level of accuracy required for simulating the effect of climate change on water quality in the runoff from northern and forested watersheds, a number of modifications were made to the model. The crop growth submodel was adapted to better represent forests, the snowmelt submodel was altered to improve the prediction of spring runoff (without requiring more input data than the original snowmelt submodel), and the evapotranspiration submodel was altered to more accurately simulate the evaporation, or sublimation, of snow. An alternate submodel for calculating rates of runoff and an alternate submodel for calculating percolation were also examined. Furthermore, a number of bugs in the model code were corrected. The changes made and the alternate submodels examined are described in this report. The changes made to the model code, including the bug fixes, are listed in the appendix. Finally, the changes in the simulations of water quality and quantity achieved by the model modifications are compared to field data.

ACKNOWLEDGMENTS

The authors wish to thank Xing Fang of the Department of Civil Engineering, Lamar University, Beaumont, TX, for writing the program for extracting the climate change data from the CCC GCM output; Jeff Arnold, Nancy Sammons of the ARS Grassland, Soil, and Water Research Laboratory, Temple, TX, and Ranjan Muttiah and R. Srinivasan of the Blackland Research Center, Texas A&M University, Temple, TX, for supplying user support and the SWAT and SWAT/GRASS code; and Omid Mohseni of the St. Anthony Falls Laboratory, Minneapolis, MN, for many helpful discussions and ideas. This study was funded in part by the U.S. Department of Agriculture, Agricultural Research Service, Grazingland Research Laboratory, El Reno, Oklahoma, in cooperation with the U.S. Environmental Protection Agency Mid-Continent Ecology Division, Duluth, Minnesota. Robert Williams, Frank Schiebe, Virginia Snarski, and John Eaton were project officers.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

I. INTRODUCTION

As part of a research project on the potential effects of climate change on aquatic ecosystems, a basin scale hydrologic model, the Agricultural Research Service's (ARS) Soil and Water Assessment Tool (SWAT), was chosen to simulate the effect of climate change on the non-point source inputs from rural watersheds to lakes and impoundments (Hanratty, 1995). The goal of this study was to develop methods for conducting a regional analysis of watershed response to climate change. Therefore, the model was applied to three different watersheds, the Baptism River near Beaver Bay, MN, the Cottonwood River near New Ulm, MN, and the Little Washita River near Ninnekah, OK. These three watersheds were chosen because they differ substantially in land use and climate conditions and because their runoff quantity and quality have been recorded. The Baptism River drains a mostly forested, 352 km^2 watershed on the north shore of Lake Superior. The Cottonwood River, a tributary of the Minnesota River, drains a 3400 km^2 agricultural watershed in south-western Minnesota. At the USGS gauging station just upstream from Ninnekah, OK, the Little Washita River drains a 538 km^2 agricultural watershed. It empties into the Washita River about 9.5 km downstream of Ninnekah.

In order to obtain the level of accuracy required for simulating the effect of climate change on water quality in these rivers, a number of modifications were made to the ARS/SWAT model. The crop growth submodel was adapted to better represent forests, the snowmelt submodel was altered to improve the prediction of spring runoff (without requiring more input data than the previous snowmelt submodel), and the evapotranspiration submodel was altered to more accurately simulate the evaporation, or sublimation, of snow. An alternate submodel for calculating runoff and an alternate submodel for calculating percolation were also examined. Furthermore, a number of bugs in the model code were corrected. The changes made and the alternate submodels examined are described in this report. The changes made to the model code, including the bug fixes, are listed in the appendix. Finally, the changes in the simulations of water quality and quantity are compared to field data.

Model Description

A model of watershed hydrology and water quality, the Soil and Water Assessment Tool, 1996 version (SWAT 96; Arnold *et al.*, 1994), was chosen from several available models (Hanratty, 1995). Other models considered included AGNPS 5.0 (Young *et al.*, 1989), HSPF (Johanson *et al.*, 1984), and SWRRB (Arnold *et al.*, 1990). Because SWAT was designed to be applied to ungaged river basins, it was anticipated

that the model could be used for analyzing many watersheds in a geographic region using data that are readily available. SWAT is a semi-empirical, distributed parameter model designed to predict the impact of management decisions on water, sediment, nutrient, and agricultural chemicals in large, engaged basins. SWAT's hydrologic predictions are based on a mass balance of the soil moisture using up to ten soil layers. It incorporates well-established ARS hydrologic methods such as the SCS Curve Number method to determine infiltration and runoff and the Modified Universal Soil Loss Equation (MUSLE) to calculate erosion. Both of these methods employ variables which the model changes to represent current soil water and land cover conditions. SWAT uses physically-based methods for the computation of percolation, lateral subsurface flow, evapotranspiration, pond and wetland hydrology, channel transmission losses, soil nutrient dynamics, soil temperature, crop growth, pesticide dynamics, and agricultural management practices. The evapotranspiration algorithm includes the effect of atmospheric CO_2 concentration on plant transpiration, which is important to include in any study of the effects of carbon dioxide-induced climate change (Hanratty and Stefan, 1997b). SWAT's groundwater algorithm is a simplified physically-based method, and the snowmelt algorithm is a degree-day method. SWAT's channel flood routing is based on Manning's equation and channel geometry. Sediment routing uses particle fall velocity, flow travel time, and a sediment delivery ratio. Nutrient and pesticide routing include chemical transformations within the stream channels in the 1996 version of SWAT; in the 1994 version, SWAT 94.2, nutrients are routed as conservative substances. Weather data (*i.e.,* minimum and maximum daily air temperature, daily solar radiation, mean daily humidity, mean daily wind speed, and daily precipitation) can be generated using statistical parameters from historical records. Alternatively, historical temperature and precipitation data can be input directly to the model. Future versions of SWAT will have the capacity to read in solar radiation data as well (for more information, see Arnold *et al.,* 1990, 1993, 1994).

II. MODIFICATIONS FOR THE BAPTISM RIVER WATERSHED

The original SWAT's ability to simulate streamflow and water quality for a northern, forested watershed was limited. The model was run using the input data calculated with the IDRISI geographic information system methods described by Hanratty and Stefan (1997a). The simulated average monthly streamflow for the period 1970 to 1990 was too low throughout most of the year (Figure 1). Simulated suspended sediment yields were at least an order of magnitude too high (Figure 2), nitrate yields were an order of magnitude too high during the spring snowmelt (Figure 3), and phosphorus yields were too high throughout most of the year (Figure 4). These discrepancies between model and field were not totally surprising. The model was originally designed for agricultural watersheds and was tested only in watersheds south of 43°N latitude (Arnold *et al.*, 1994; Arnold and Allen, 1996). By applying SWAT to the

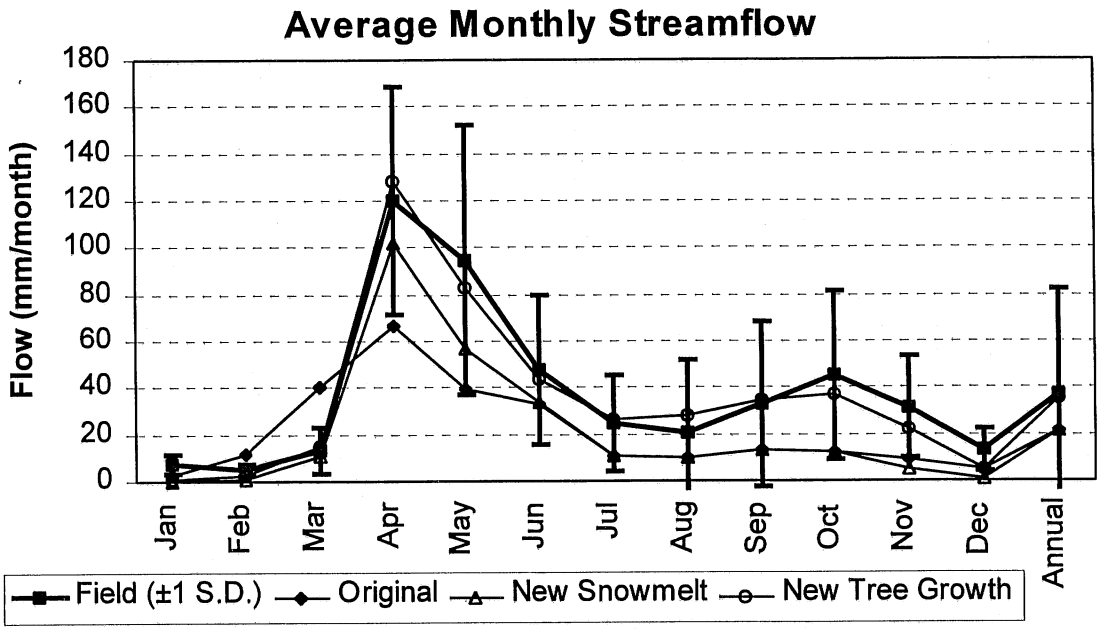


Figure 1 Comparison of the average monthly streamflow in the Baptism River near Beaver Bay, MN for the period 1970 to 1990 to the average monthly water yield simulated by SWAT 94.2 with and without the improvements to the snowmelt and tree growth submodels. "Original" corresponds to the old version in Figs. 2-4, "New tree growth" corresponds to the new version.

Baptism River watershed, a forested watershed at 48°N latitude, the intended purpose of the model was obviously being stretched. Upon further investigation and after discussion with the model's authors, it became obvious that the snowmelt model was too simplistic to represent the process of snowmelt in the Baptism River watershed and that the crop growth submodel needed to be adapted to better represent forests.

Most available snowmelt models (Osborn *et al.*, 1982) use variations of the energy balance method first developed by Wilson (1941). The amount of data required to represent all of the heat fluxes in the energy balance is a major drawback to these models. Snowmelt predictions by the degree-day method in conjunction with a suitable runoff model and a melt rate that is varied to represent changes in the snow pack are nearly as accurate as those by the process-based energy balance model (Rango and Martinec 1995). Therefore, to minimize the data requirements for the snowmelt model, a degree-day method that non-linearly increases the melt rate as more of the snow pack melts (Kim *et al.*, 1974) was employed. This method assumes that the melt rate of snow, M, is related to temperature as expressed by the relationship given in Eq. (1).

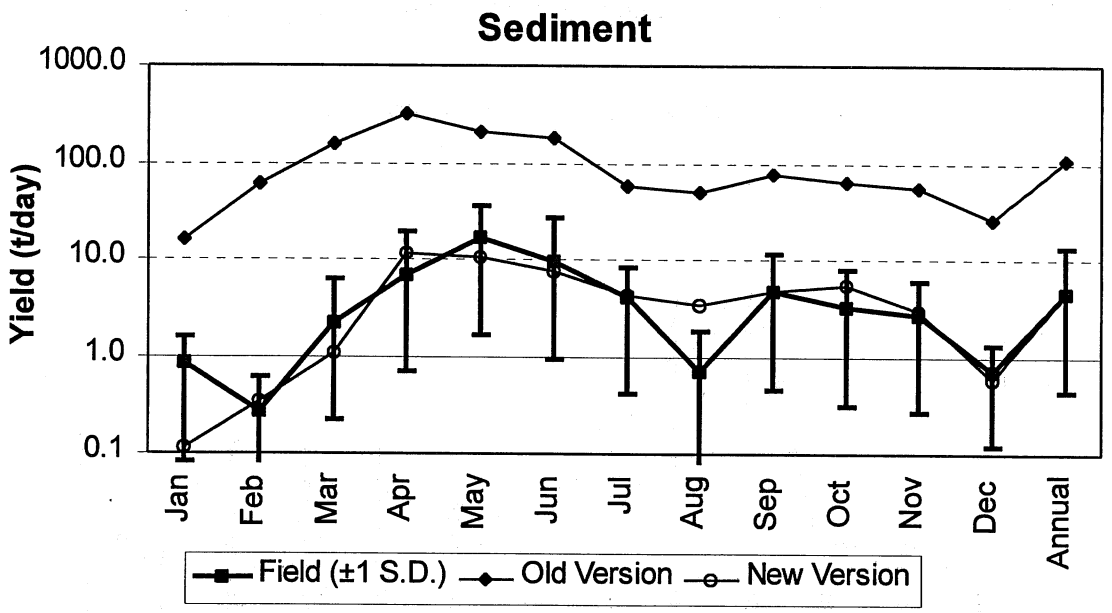


Figure 2 Model to field comparison of the average monthly sediment yield in the Baptism River near Beaver Bay, MN for 1970 to 1990 using SWAT 94.2 with (new version) and without (old version) modification and calibration.

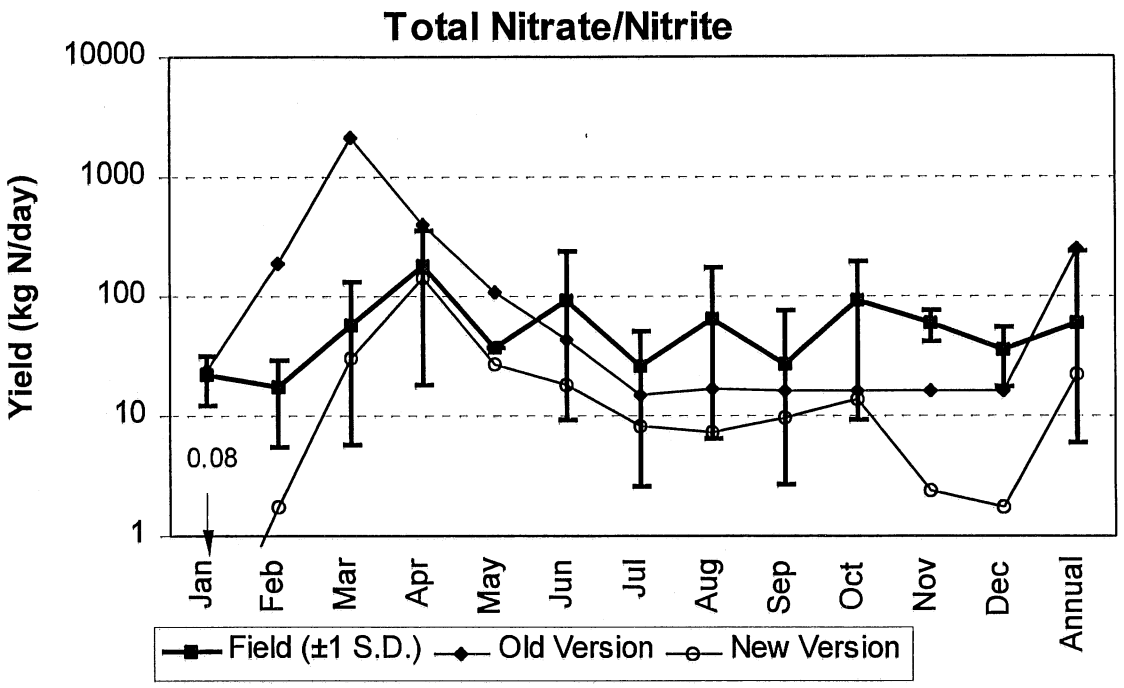


Figure 3 Model to field comparison of the average monthly nitrate/nitrite yield in the Baptism River near Beaver Bay, MN for 1970 to 1990 using SWAT 94.2 with (new version) and without (old version) modification.

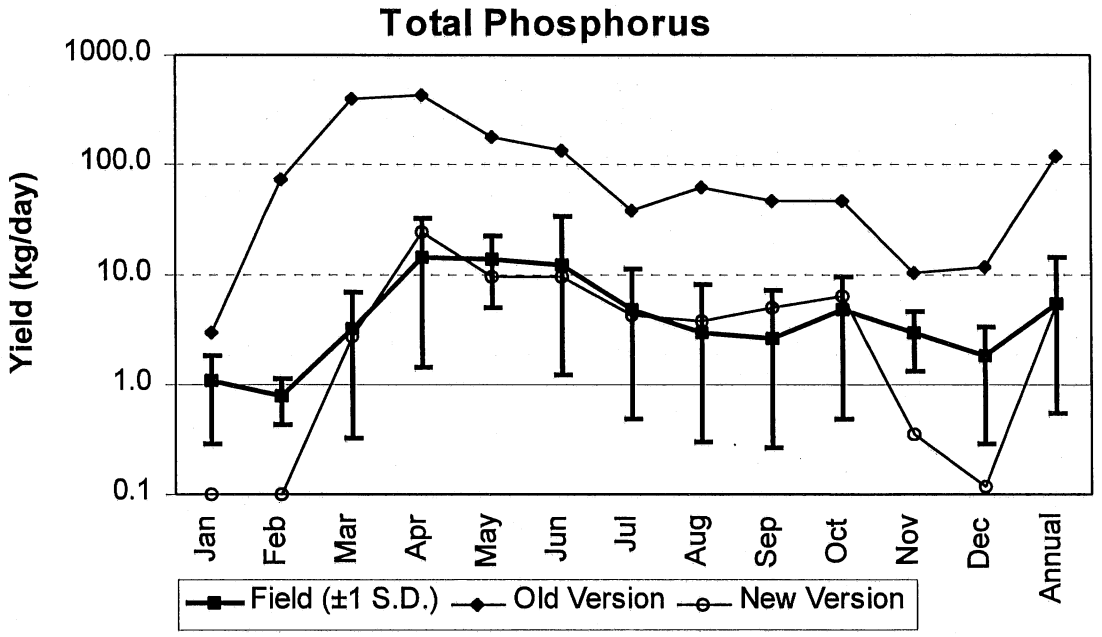


Figure 4 Model to field comparison of the average monthly phosphorus yield in the Baptism River near Beaver Bay, MN for 1970 to 1990 using SWAT 94.2 with (new version) and without (old version) modification and calibration.

$$M = a * (T - T_b) \tag{1}$$

where M = snowmelt (mm / day)

a = melt rate (mm/° C day)

T = maximum daily temperature (° C)

T_b = base temperature (° C)

The base temperature was assumed to be 0° C. The maximum daily temperature, T, includes the effect of solar radiation, the main energy source for melting of the snow pack. The melt rate, a, varies depending on the percentage of the snow pack that has already melted. An empirical relationship for a is (Kim *et al.,* 1974)

$$\begin{aligned} a &= 0.4572 && m \le 10 \\ a &= 0.005486 * m^2 - 0.118872 * m + 1.09728 && 10 < m < 35 \\ a &= 3.6576 && m \ge 35 \end{aligned} \tag{2}$$

where m = percentage of snowpack already melted.

The percentage of the snow pack that has already melted is a surrogate for changes in albedo, snow density, and snow temperature that speed the melting of the snow pack as the snowmelt season progresses.

Another change necessary was in the SWAT 94.2 data input methods for the soil data. The code from SWAT 96 was used to add a 1 cm thick top soil layer and then move the remaining layers down 1 cm to accommodate this new layer. The new first layer is given the same characteristics as the original top layer. According to the model documentation (Arnold *et al.,* 1994), it is assumed that the top soil layer is 1 cm thick.

The final change necessary was in the crop-growth submodel's methods for simulating a forest. In the original SWAT model, forests were represented only by the leaves of the trees. This is a reasonable approach, since of all the parts of a plant, the leaf area has the largest role in determining evapotranspiration. In the model, however, soil erosion and soil moisture are directly influenced by the biomass of the crop cover. Leaving out the trunk, branch, and twig biomass greatly underestimates the biomass of a forest. Therefore, SWAT's crop-growth submodel was modified to represent the biomass of the entire tree. Since the leaves were the only part of the tree simulated, the model originally simulated the loss of leaves in the fall as a kill of the crop at the end of the growing season. For mixed forests and deciduous forests, the modified model now simulates leaf loss in the fall using the "harvest only" option of the crop growth model. The model is instructed via the crop management input file to "harvest" the trees on October 15th of each year. The leaf biomass, which is calculated as three percent of the

tree biomass (Schmitt and Grigal, 1981; Koerper and Richardson, 1980), is subtracted from the crop biomass and added to the residue ground cover. The leaf area index is decreased by 90% for a mixed forest and by 100% for a deciduous forest. The initial biomass of the forest (kg/ha) was estimated from Miles *et al.*(1990). Furthermore, to simulate the deep root structure of trees, the nutrient uptake algorithm was modified so that no nutrients are withdrawn from the 1 cm thick top soil layer by the trees. After the changes were made to the computer code and the crop management input file, the parameters for a mixed forest in SWAT's crop database were adjusted to represent the mixed forest on the North Shore of Lake Superior. The harvest index was set to 0.03 to represent the leaf biomass and the maximum leaf area index (LAI) was set to 3.5 to represent a forest with more deciduous trees (maximum LAI = 2.0) than coniferous trees (maximum LAI = 6.0). The exact value of the maximum LAI was chosen by calibration to the mean monthly streamflow.

The standard errors, the coefficients of variation (C.V. = standard error as a percent of the field mean), and the goodness of fit for mean monthly streamflow, sediment yield, phosphorus yield, and nitrogen yield for the years 1970 to 1990 (the years for which water quality data were available) are shown in Table 1 for both the original SWAT 94.2 and the modified versions. The model improvements significantly reduced the errors in the model output. The phosphorus and sediment yield simulations improved the most (the standard error decreased by a factor of 15 and 47, respectively) and the modified SWAT's simulation of mean monthly streamflow was more accurate than the original one (25.8 mm/month vs. 7.6 mm/month standard error). Plots of simulated and measured streamflow, sediment yield, nitrogen yield, and phosphorus yield for the period 1970 to 1990 are shown in Figs. 1, 2, 3, and 4, respectively. The simulated mean monthly streamflow was quite accurate throughout the entire year (Figure 1). The simulated mean monthly sediment yield was accurate throughout most of the year, but was too high in August (Figure 2). The modifications made to SWAT improved the simulation of mean monthly nitrogen yield (Figure 3). Both the magnitude and the timing of the spring peak were much better. The simulated nitrogen values during the rest of the year, however, were still consistently lower than the observed values. The accuracy of the simulated mean monthly phosphorus values was dramatically improved (Figure 4). The original SWAT was consistently too high, where the modified SWAT accurately simulated the spring peak and was within one standard deviation of the field data in eight months of the year and for the average annual phosphorus yield.

The changes to SWAT improved the simulations of mean monthly water yield by making more accurate estimates of the timing of the snowmelt and of the evapotranspiration in the Baptism River watershed's forest. The new snowmelt submodel was able, without requiring more input data, to predict the amount and the timing of the watershed's average spring melt. The new approach to modeling tree growth required more input data than the original method, but decreased the average evapotranspiration simulated by the model, which produced a more accurate simulation of the streamflow. There are two limitations of this new tree-growth submodel. First, the

III. MODIFICATIONS FOR THE LITTLE WASHITA RIVER WATERSHED

The input data for the Little Washita River watershed were calculated using SWAT/GRASS (Srinivasan and Arnold, 1994) as described by Hanratty and Stefan (1997c). SWAT96 was run to simulate the period 1963 to 1979. SWAT modifies the condition II curve number (*i.e.*, the input parameters for the SCS curve number method for computing the rate runoff) to represent the current soil water condition. The algorithm in the code for altering the curve number, however, did not match the algorithm in the model documentation (Arnold *et al.*, 1994). The coded algorithm used the amount of soil water (in mm) from all of the soil layers to calculate the change in the curve number as shown in equation 3:

$$s = s_{\max}\left(1 - \frac{sum}{sum + \exp(w_1 - w_2\, sum)}\right) \tag{3}$$

where s = SCS Curve Number Method retention parameter

$s_{\max}$ = maximum value of s for this land cover and soil type

w_1, w_2 = shape parameters calculated from soil properties

$$sum = \sum_{i=1}^{M} SW_i$$

SW_i = soil water in layer i (mm)

M = number of soil layers

The documented algorithm used a depth weighted sum of the fraction of field capacity in the soil layers within one meter of the surface; equation 4 is used by the code to calculate the value of *sum* in equation 3.

$$sum = \frac{\sum_{i=1}^{M} FFC_i \frac{Z_i - Z_{i-1}}{Z_i}}{\sum_{i=1}^{M} \frac{Z_i - Z_{i-1}}{Z_i}}, \quad (4)$$

where $FFC_i = \frac{SW_i - WP_i}{FC_i - WP_i}$

SW_i = soil water in layer i (mm)

WP_i = wilting point for layer i (mm)

FC_i = field capacity for layer i (mm)

Z_i = depth at the bottom of layer i (mm)

The algorithm in the code for simulating crack flow also differed from the documented algorithm. It calculated crack flow rates using the saturated conductivity of the soil, rather than the soil clay content. The algorithm in the code is shown in equation 5.

$$O_i = O_{i-1}\left[1 - \exp\left(\frac{-24\zeta_i}{O_{i-1}}\right)\right] \quad (5)$$

where O_i = flow from layer i to the layer below

O_{i-1} = flow to layer i from the layer above

$\zeta_i = 0.003 SC_i$

SC_i = saturated conductivity of layer i

In the documented algorithm, the value of ζ_i in equation 3 is calculated from soil water and clay content using equation 4.

$$\zeta_i = 0.01\, CLA_i\left(1 - \frac{SW_i}{SW_i + \exp(7.0 - 0.22\, SW_i)}\right) \quad (6)$$

In search of a method to accurately simulate streamflow without calibration, the documented algorithms were returned to the code. One of the coefficients for the documented crack flow equation, however, needed to be calibrated in order to yield an acceptable representation of the streamflow during wet years. Therefore, the original, non-documented algorithms were returned to the code and the condition II curve numbers were calibrated to yield the best fit between simulated and measured streamflow. The curve numbers were decreased by 20% from the values calculated by SWAT/GRASS to achieve a best fit.

The monthly streamflow simulations are compared to the field data in Fig. 5. The monthly simulated streamflow was most inaccurate when using the non-calibrated, non-documented algorithms (standard error = 9.96 mm, Nash-Sutcliffe = -3.25). The best accuracy was when the non-documented algorithms were calibrated. While the non-calibrated, documented algorithms produced better simulations than the non-calibrated, non-documented algorithms, the calibration results were not as accurate as with the non-documented algorithms. In particular, the documented algorithm's monthly streamflow simulations were too high when heavy rains occurred in September or October after a dry summer (1970-1973). These inaccuracies would have led to very large errors in the simulation of climate change effects on runoff, since the global circulation model projected a large increase in average precipitation during October under a $2xCO_2$ scenario. (Hanratty and Stefan, 1997c). Therefore, the non-documented algorithms that were in SWAT were calibrated and used for simulating the effect of climate change on streamflow and water quality.

Table 2 Average basin values for rainfall, simulated surface runoff, and simulated subsurface flow that contributes to streamflow in the Cottonwood River at New Ulm, MN.

Month	Rainfall (mm)	Surface Q (mm)	Sub-surface Q (mm)
1	17.64	0.0178	0.0004
2	16.61	1.081	0.0016
3	46.76	11.44	0.0353
4	69.79	27.57	0.3265
5	75.89	7.711	0.3965
6	97.73	10.36	0.5311
7	84.48	4.447	0.1258
8	75.42	2.821	0.0128
9	73.09	3.563	0.0247
10	56.44	4.420	0.0390
11	38.89	1.370	0.0225
12	20.44	0.0823	0.0057

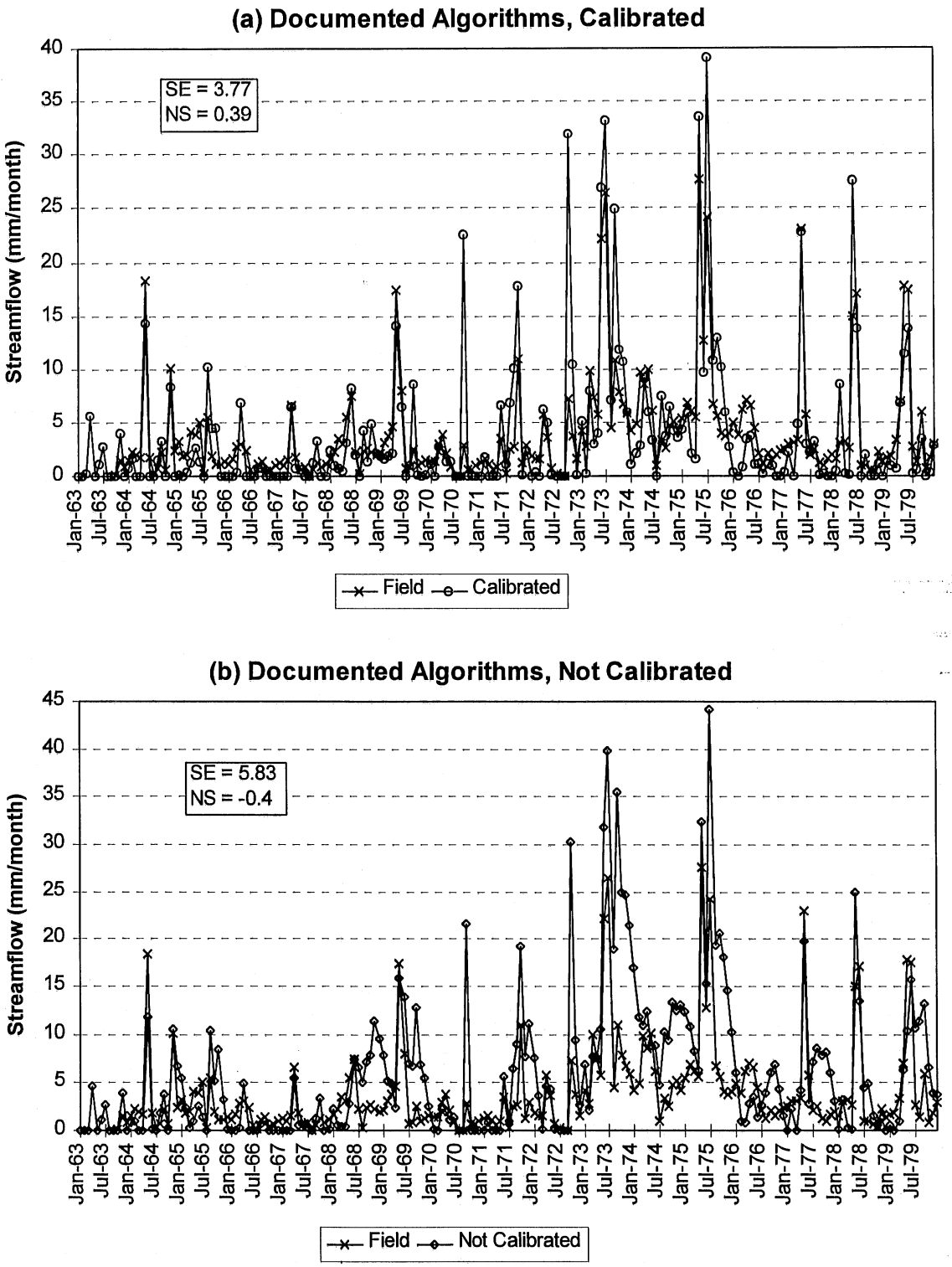
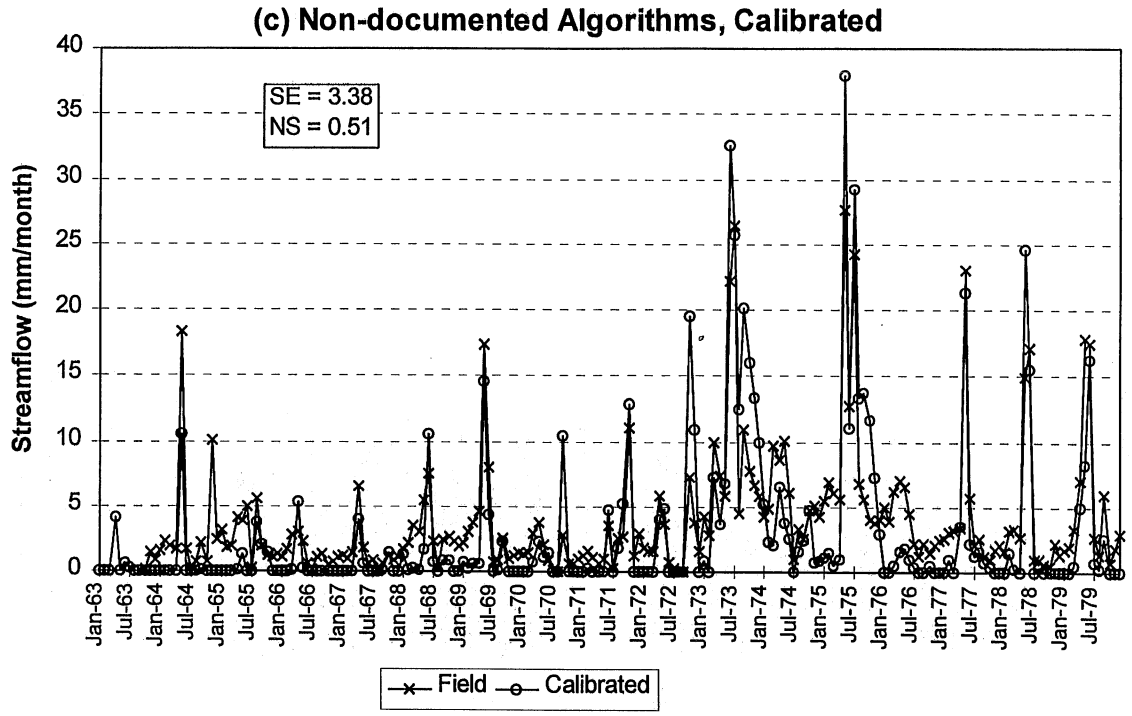


Figure 5 Model to field comparison of the monthly streamflow in the Little Washita River near Ninnekah, OK. Calibrated (a) and non-calibrated (b) simulations from SWAT96 are compared with field data using the documented algorithms for direct runoff and flow through cracks in the soil.



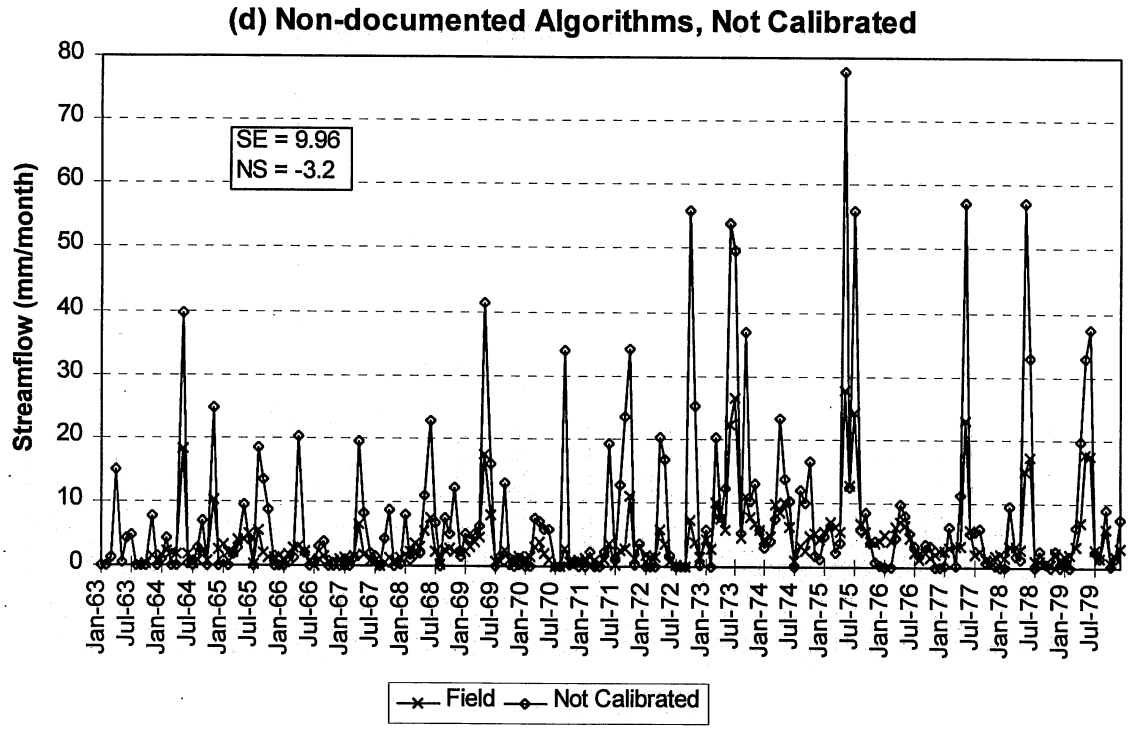


Figure 5 **(cont'd.)** Model to field comparison of the monthly streamflow in the Little Washita River near Ninnekah, OK. Calibrated (c) and non-calibrated (d) simulations from SWAT96 are compared with field data using the non-documented algorithms for direct runoff and flow through cracks in the soil.

IV. MODIFICATIONS FOR THE COTTONWOOD RIVER WATERSHED

The two documented algorithms tested on the Little Washita River watershed, the runoff curve number algorithm and the crack flow algorithm, did not work in the Cottonwood River watershed simulations, either. The documented algorithms produced simulations of the Cottonwood River streamflow that were too high during the winter and fall. Both of the non-documented algorithms needed to be returned to the model and the snowmelt algorithm needed adjustment in order to reduce the streamflow enough to match the measured streamflow during those times of the year (Fig. 6). Snowmelt for the Cottonwood River watershed was calculated using the average daily temperature instead of the maximum daily temperature.

Mean monthly streamflow, however, was still too low during the spring (Fig. 6). It was hypothesized that the reason for the discrepancy was that the model was not representing the extensive tile drainage systems in the watershed. Therefore, the lateral and vertical hydraulic conductivities of the bottom soil layer (0.5 to 1.5 m below the surface) were increased and decreased, respectively, by a factor of 1000 for each cropland

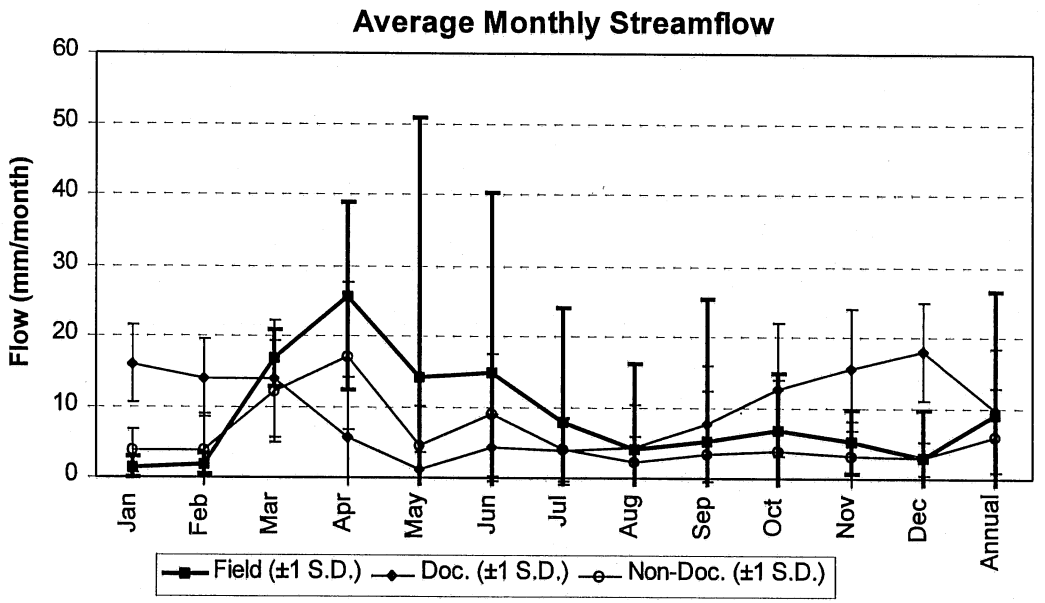


Figure 6 Model to field comparison of the average monthly streamflow in the Cottonwood River near New Ulm, MN. SWAT96 simulations using the documented and non-documented algorithms for direct runoff and for flow through cracks in the soil are compared with field data.

subbasin that was poorly drained (soil hydraulic saturated conductivity less than 50 mm/hr). This change had very little effect on the mean monthly streamflow because most of the simulated streamflow for the Cottonwood River originated from surface runoff, rather than sub-surface flow (Table 2). Therefore, the changes to the hydraulic conductivities to represent tile drainage were removed from the model after the unsuccessful test.

The next hypothesis was that the simulated streamflow was too small in the spring because the simulated accumulation of snow was too small. Therefore, the algorithm for calculating potential evaporation from the snowpack was modified. SWAT uses the Penman-Monteith method for calculating potential evapotranspiration. The potential soil evaporation, E_O, is then multiplied by a soil cover index, *EA,* that is calculated based on crop biomass and residue. The model documentation (Arnold, *et al.,* 1994) states that *EA* is set to 0.5 when the snow water content is 5 mm or more. The model code, however, did not change *EA* when there was snow on the ground. Snow evaporation (sublimation) was too high when *EA* was set to 0.5, so a calibration coefficient, EA_S, was added. Potential evaporation (adjusted for cover), E_S, when there is 5 mm or more water content of snow on the ground is then calculated by:

$$E_s = E_o \cdot EA \cdot EA_s \tag{7}$$

The calibrated value of the constant EA_S for the Cottonwood River basin, 0.01, provided a better fit to the spring peak runoff (April). Sublimation of snow and soil evaporation, therefore, was quite small when there is a snow cover of 5 mm water equivalent or more (~50 to 100 mm of snow). In northern climates, sublimation is small for most of the winter because the air temperature is lower than the snowpack temperature (Anderson, 1967).

As a calibration measure, the SCS condition II curve numbers for all of the subbasins were then increased by 5% to improve the overall fit of the simulated streamflow. The resulting streamflow simulation is compared with field data in Fig. 7. The standard error and Nash-Sutcliffe coefficient for the mean monthly streamflow were 3.31 mm and 0.78, respectively. Before calibrating snow evaporation and the curve numbers, they were 4.71 mm and 0.56, respectively. For monthly streamflow, the standard error and Nash-Sutcliffe coefficient were 9.67 mm and 0.62, an improvement over the pre-calibration values of 10.89 mm and 0.52, respectively.

SWAT was also run to simulate the Baptism River watershed using the new snow evaporation algorithm. It did not change the simulated streamflow significantly; the change in standard error was about 0.01 mm. The forested Baptism River watershed, however, required that the snowmelt algorithm use the maximum daily air temperature rather than the average air daily temperature. When the average daily temperature was used to determine snowmelt, the peak in average monthly streamflow was one month too late. The net radiation (*i.e.,* energy) for available for snowmelt, then, was better represented in the Baptism River watershed by the maximum daily air temperature.

In contrast, snowfall and snowmelt is such a small part of the Little Washita River's water budget that these changes to the snowmelt and snow evaporation algorithms made no difference in the simulated monthly streamflow.

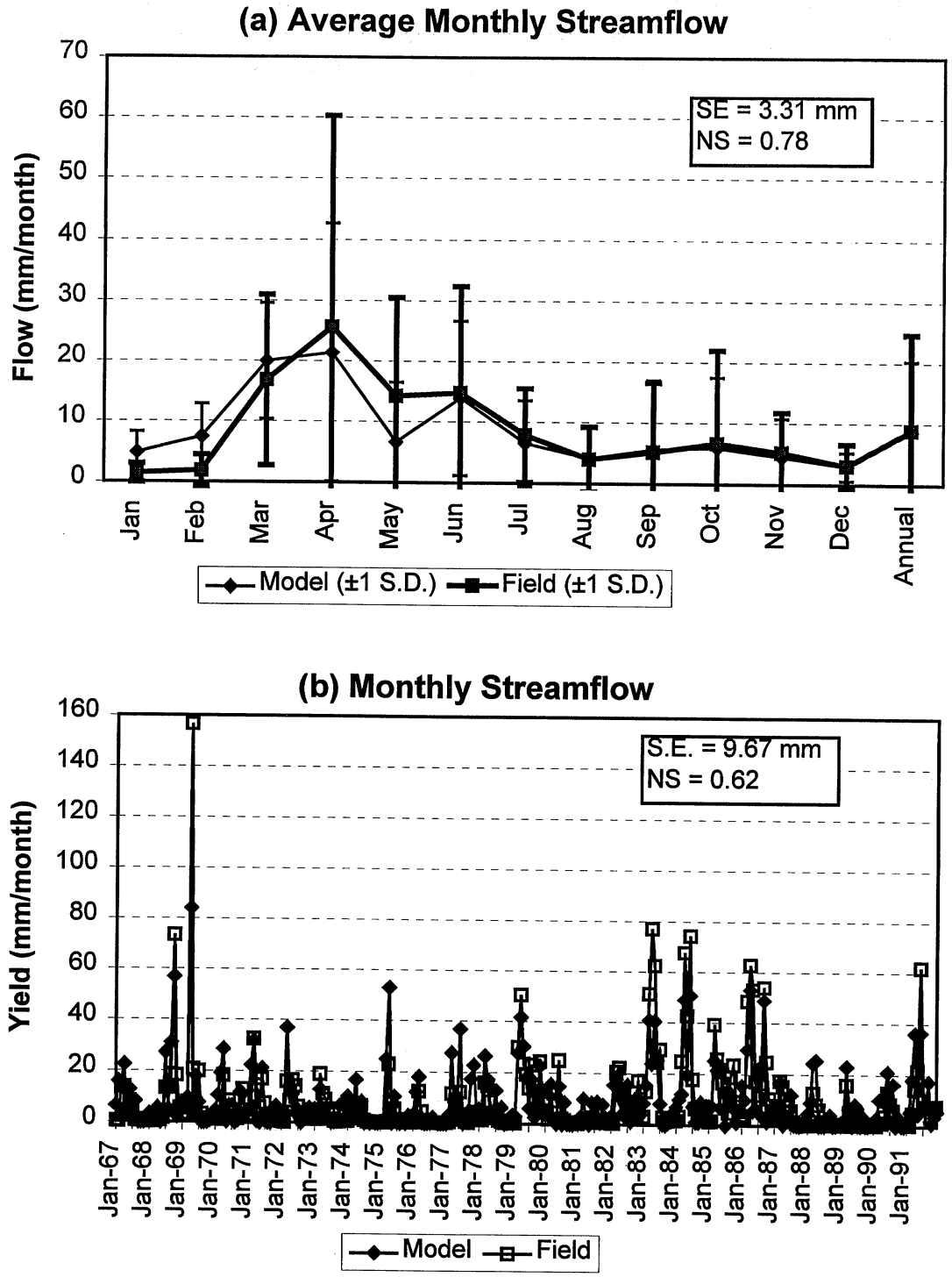


Figure 7 Model to field comparison of the (a) average monthly streamflow and (b) monthly streamflow in the Cottonwood River near New Ulm, MN, after modification and calibration of SWAT96.

V. COMPARISON

The improvements in the simulated water quantity and quality described above include the improvements due to calibration as well as those due to modification of the model's algorithms and to corrections of bugs in the computer code. The streamflow and water quality predictions from the original code obtained from the model creators (Arnold, pers. comm.) and from the new version developed as part of this research project were therefore compared for the Baptism, Cottonwood, and Little Washita River watersheds. The exact same input data were used for each version of the model. Any differences between the "old" version and the "new" version, therefore, are due to changes made in the model's computer code, and not to the calibrated values of the input parameters. The results for the Cottonwood (1967-1991), Baptism (1970-1990), and Little Washita (1963-1979) are shown in Figs. 8, 9, and 10, respectively. The monthly average streamflow, nitrate yield, total phosphorus yield, sediment yield, and ammonia/organic nitrogen yield simulated by the old version, simulated by the new version, and measured at the basin outlet are compared. Since the three nutrients were measured in the Little Washita River for only the period October 1967 to September 1969, the mean measured values are not included in Fig. 10, but the monthly measured values for nitrate, phosphorus, and ammonia/organic N during that time period are compared to both model versions in Fig. 11. Among all three watersheds, there were only two cases where the new version was not clearly a better representation of the measured values: nitrate/nitrite in the Baptism River (Fig. 9) and ammonia/organic N in the Little Washita River (Fig. 11).

Overall, the new version provided a better simulation of the measured conditions. The changes made in the FORTRAN code to create the new version are shown in the Appendix.

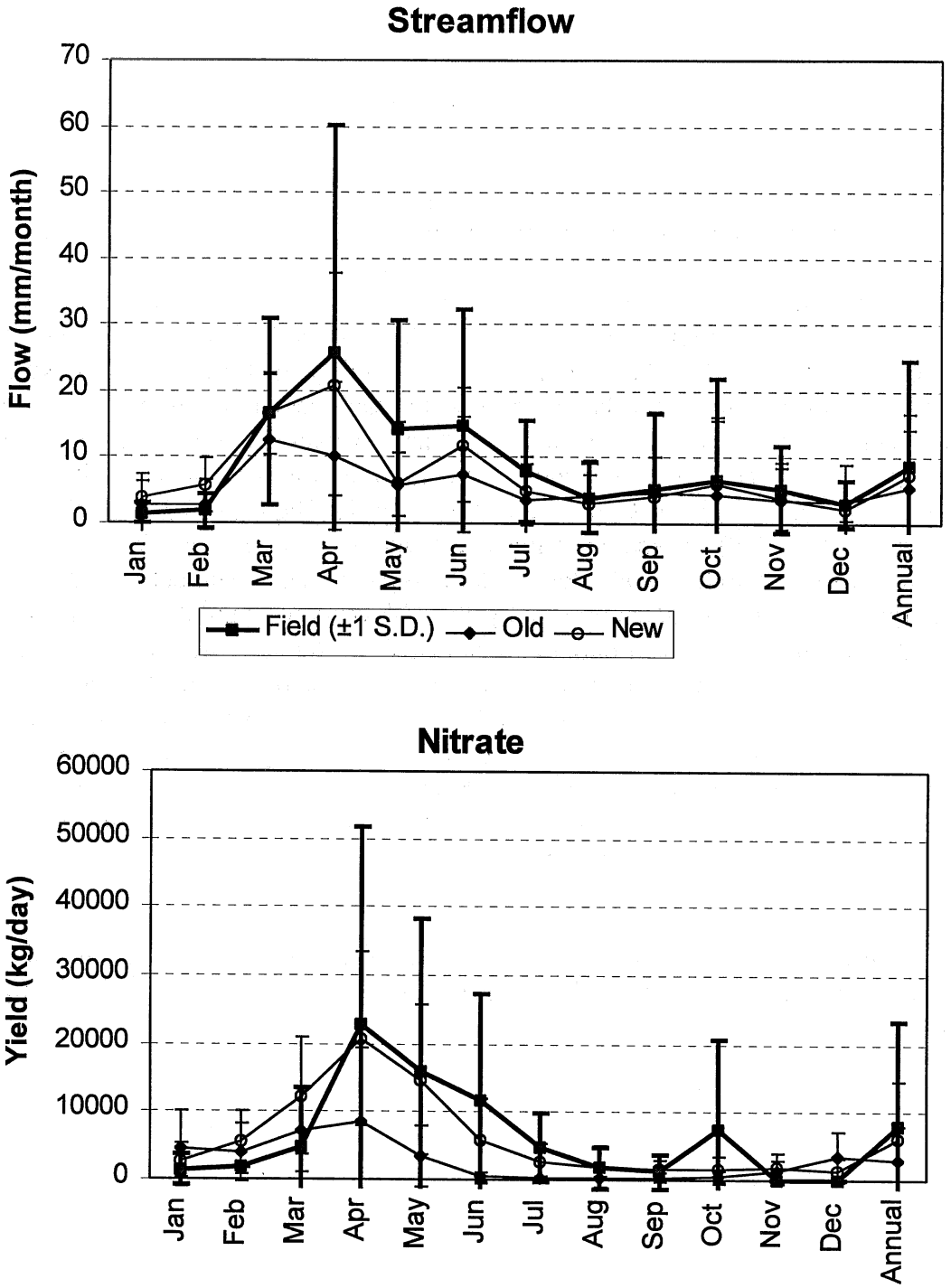


Figure 8 Comparison of simulated average monthly streamflow, nitrate yield, total phosphorus yield, sediment yield, and ammonia/organic N yield from the original SWAT96 (Old) and the modified version (New) to data measured in the Cottonwood River near New Ulm, MN, from 1967 to 1991 using the same (calibrated) input data for both versions.

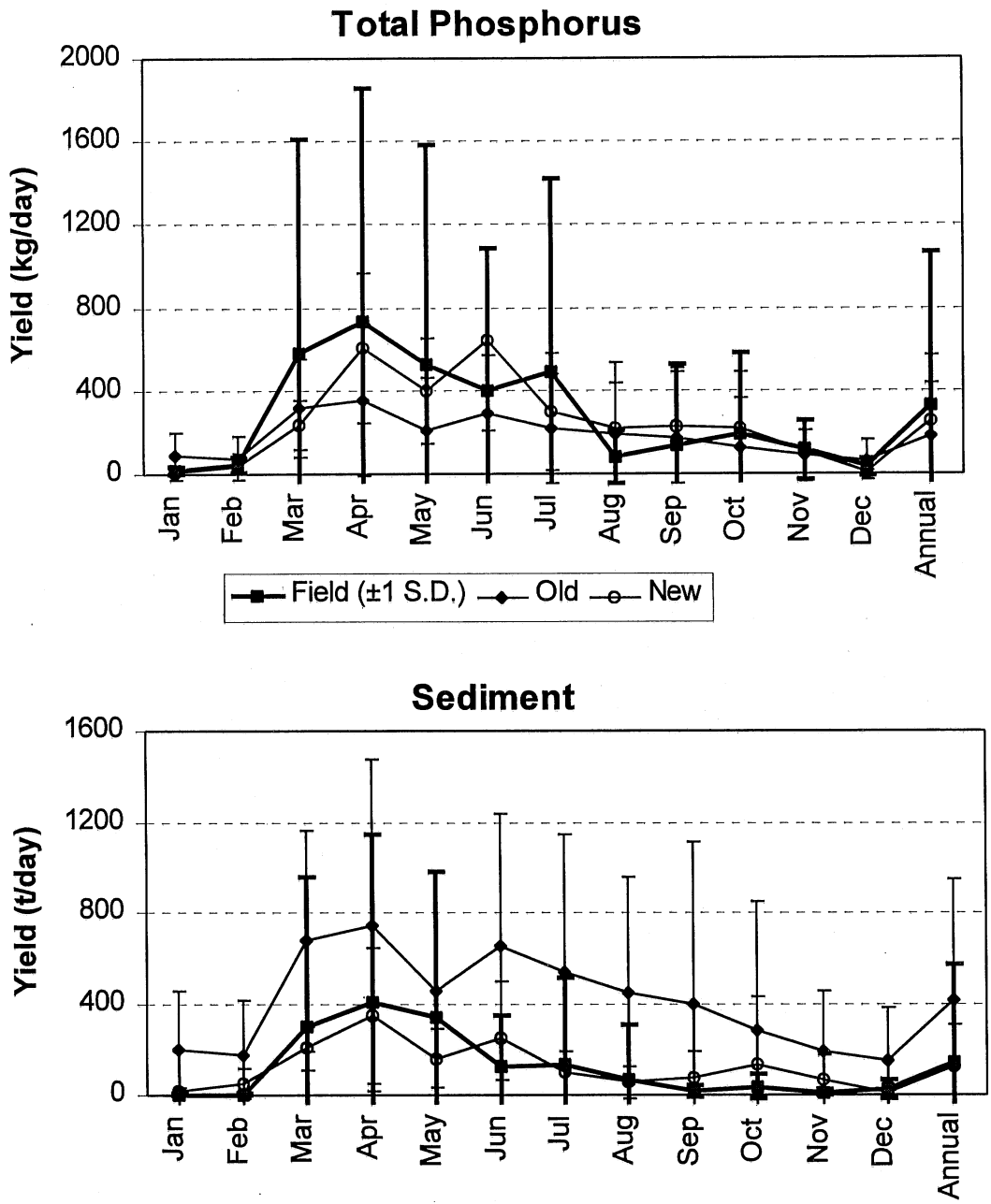


Figure 8 **(Cont'd.)** Comparison of simulated average monthly streamflow, nitrate yield, total phosphorus yield, sediment yield, and ammonia/organic N yield from the original SWAT96 (Old) and the modified version (New) to data measured in the Cottonwood River near New Ulm, MN, from 1967 to 1991 using the same (calibrated) input data for both versions.

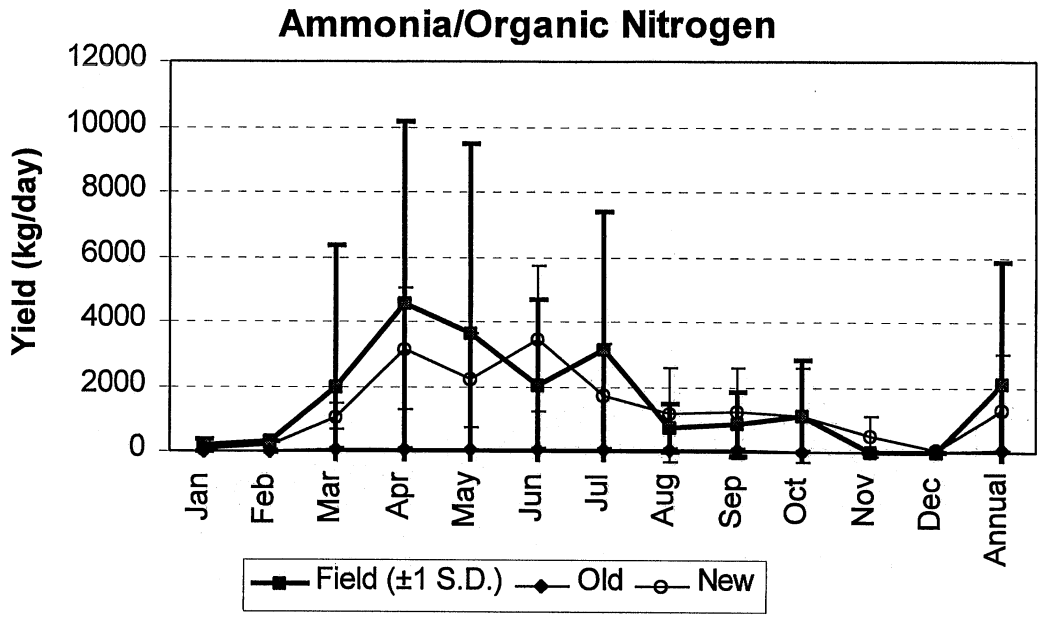


Figure 8 **(Cont'd.)** Comparison of simulated average monthly streamflow, nitrate yield, total phosphorus yield, sediment yield, and ammonia/organic N yield from the original SWAT96 (Old) and the modified version (New) to data measured in the Cottonwood River near New Ulm, MN, from 1967 to 1991 using the same (calibrated) input data for both versions.

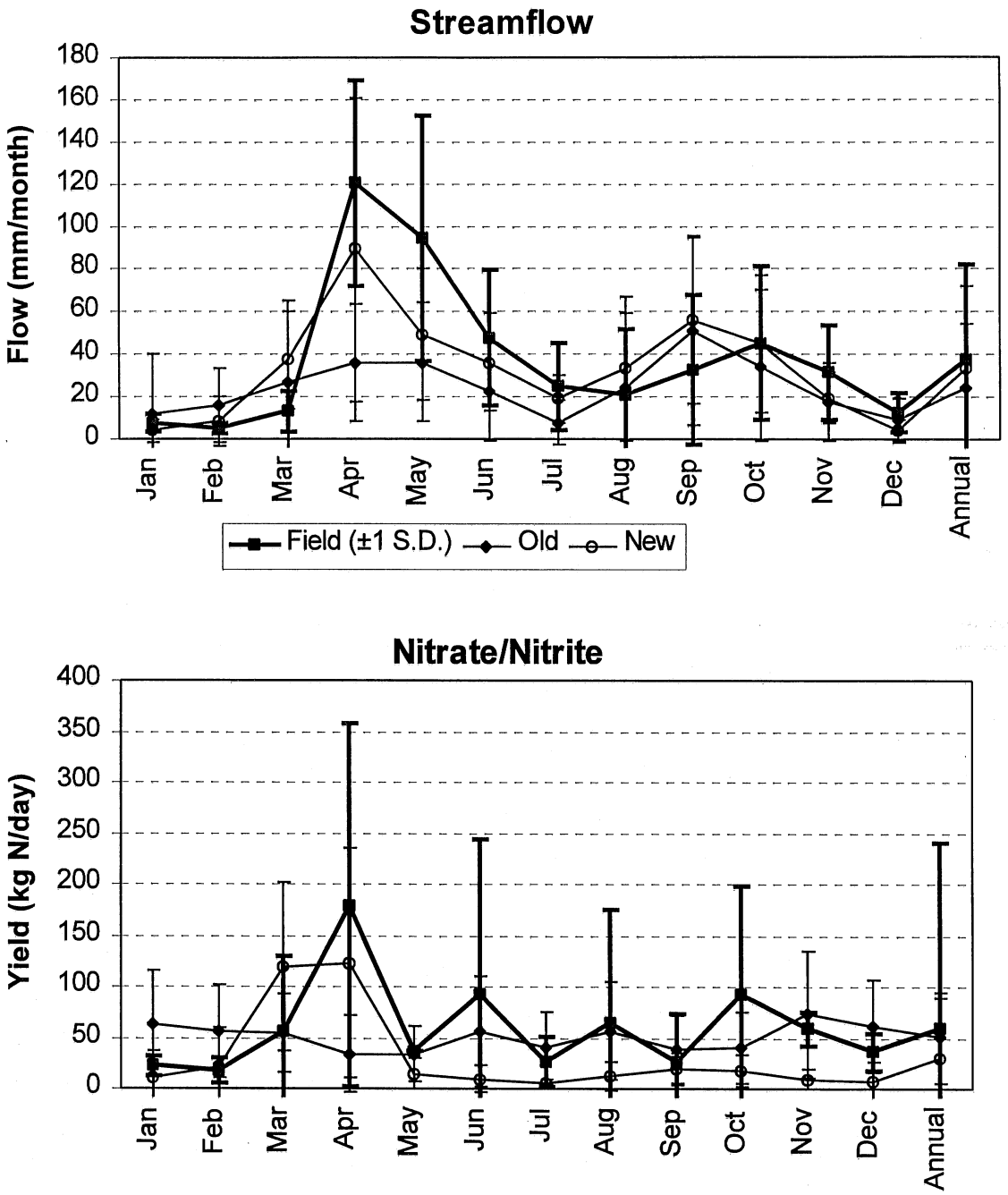
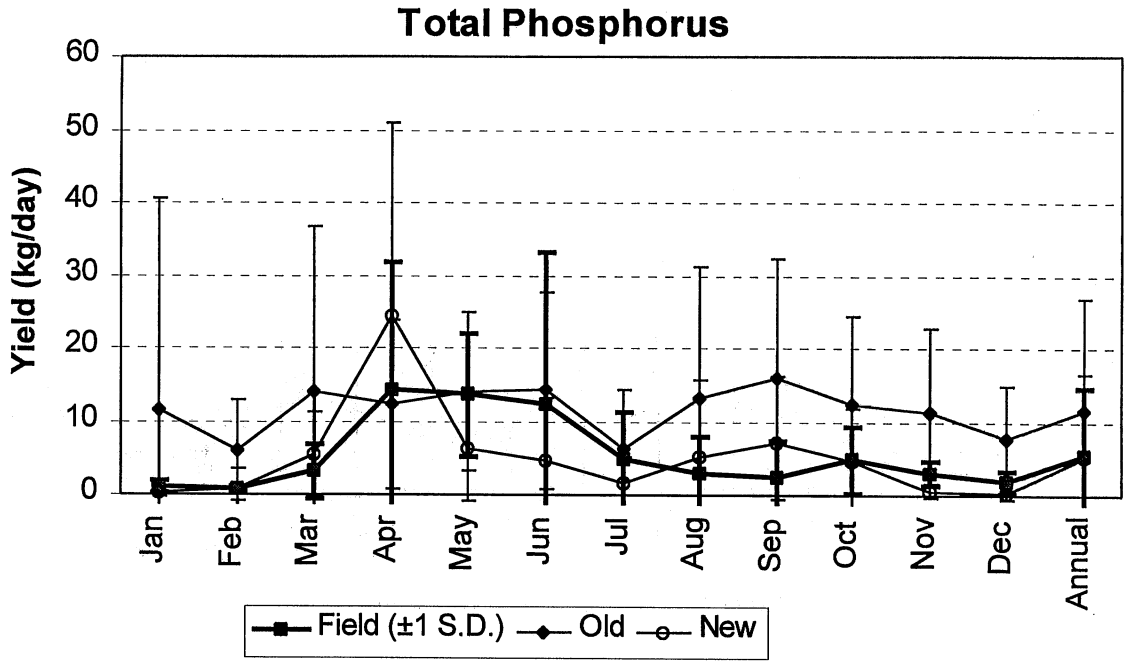


Figure 9 Comparison of simulated average monthly streamflow, nitrate yield, total phosphorus yield, sediment yield, and ammonia/organic N yield from the original SWAT96 (Old) and the modified version (New) to data measured in the Baptism River near Beaver Bay, MN, from 1970 to 1990 using the same (calibrated) input data for both versions.



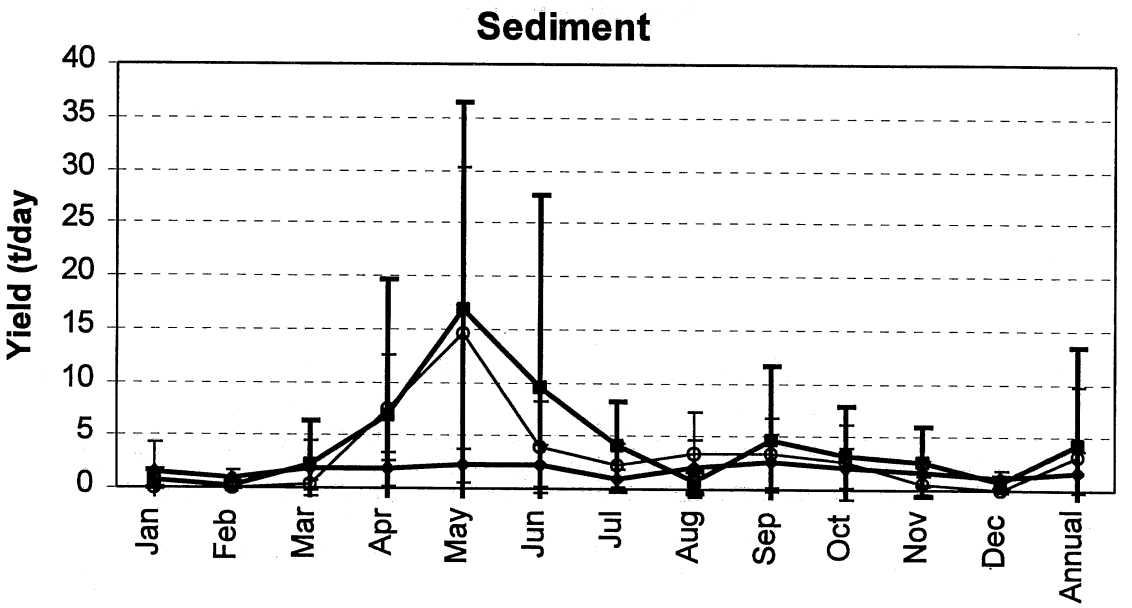


Figure 9 **(cont'd)** Comparison of simulated average monthly streamflow, nitrate yield, total phosphorus yield, sediment yield, and ammonia/organic N yield from the original SWAT96 (Old) and the modified version (New) to data measured in the Baptism River near Beaver Bay, MN, from 1970 to 1990 using the same (calibrated) input data for both versions.

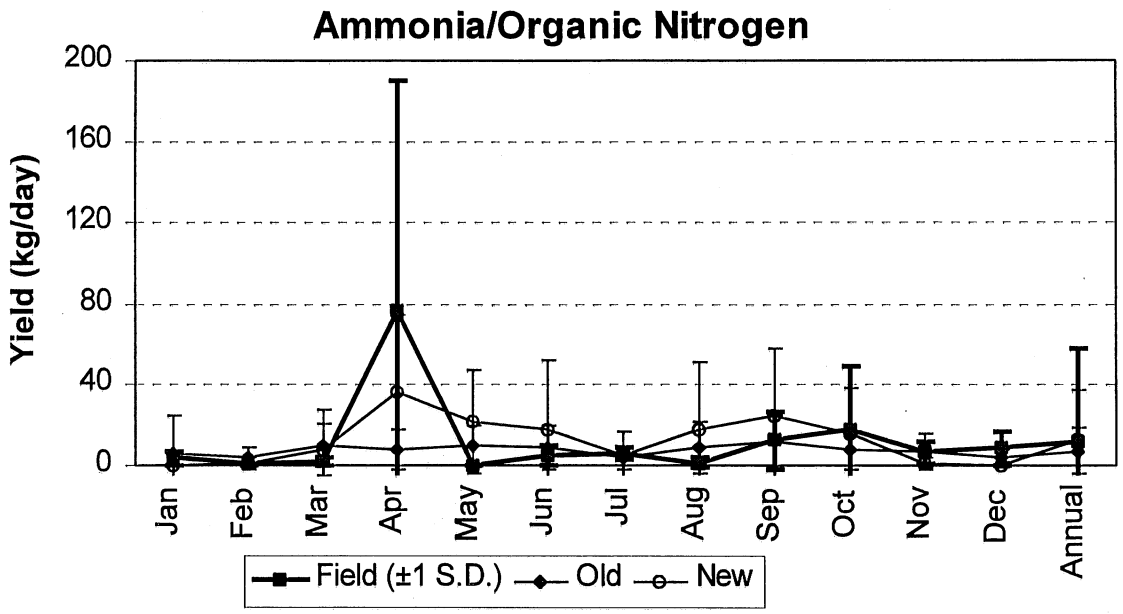


Figure 9 **(cont'd)** Comparison of simulated average monthly streamflow, nitrate yield, total phosphorus yield, sediment yield, and ammonia/organic N yield from the original SWAT96 (Old) and the modified version (New) to data measured in the Baptism River near Beaver Bay, MN, from 1970 to 1990 using the same (calibrated) input data for both versions.

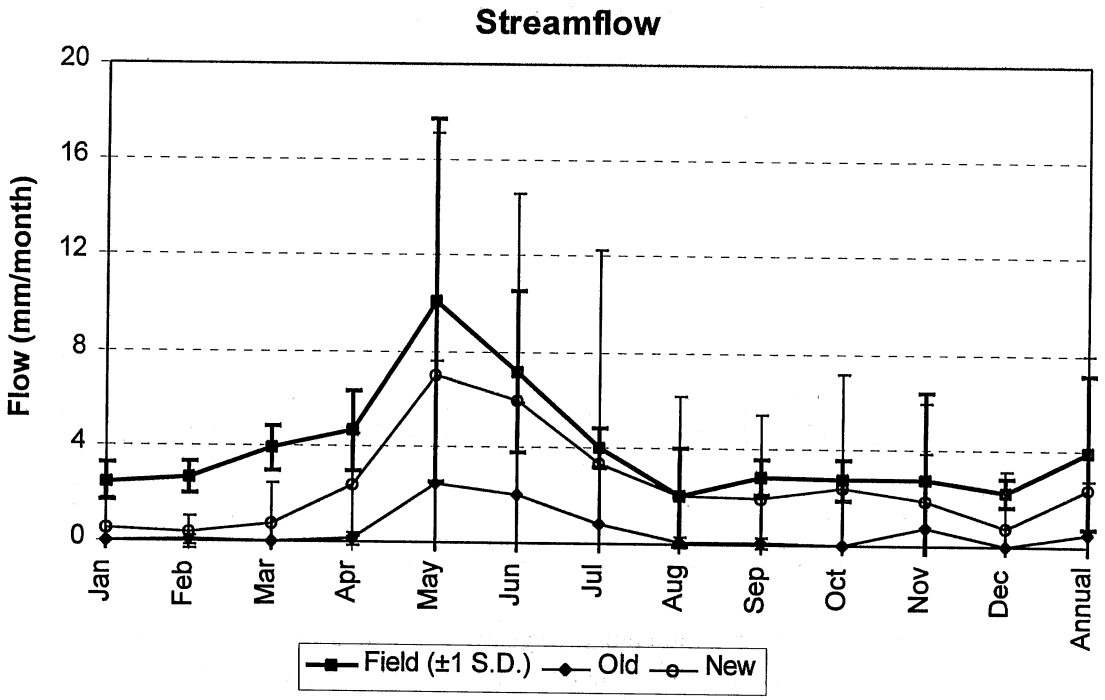


Figure 10 Comparison of simulated average monthly streamflow, nitrate yield, total phosphorus yield, sediment yield, and ammonia/organic N yield from the original SWAT96 (Old) and the modified version (New) to data measured in the Little Washita River near Ninnekah, OK, from 1963 to 1979 using the same (calibrated) input data for both versions.

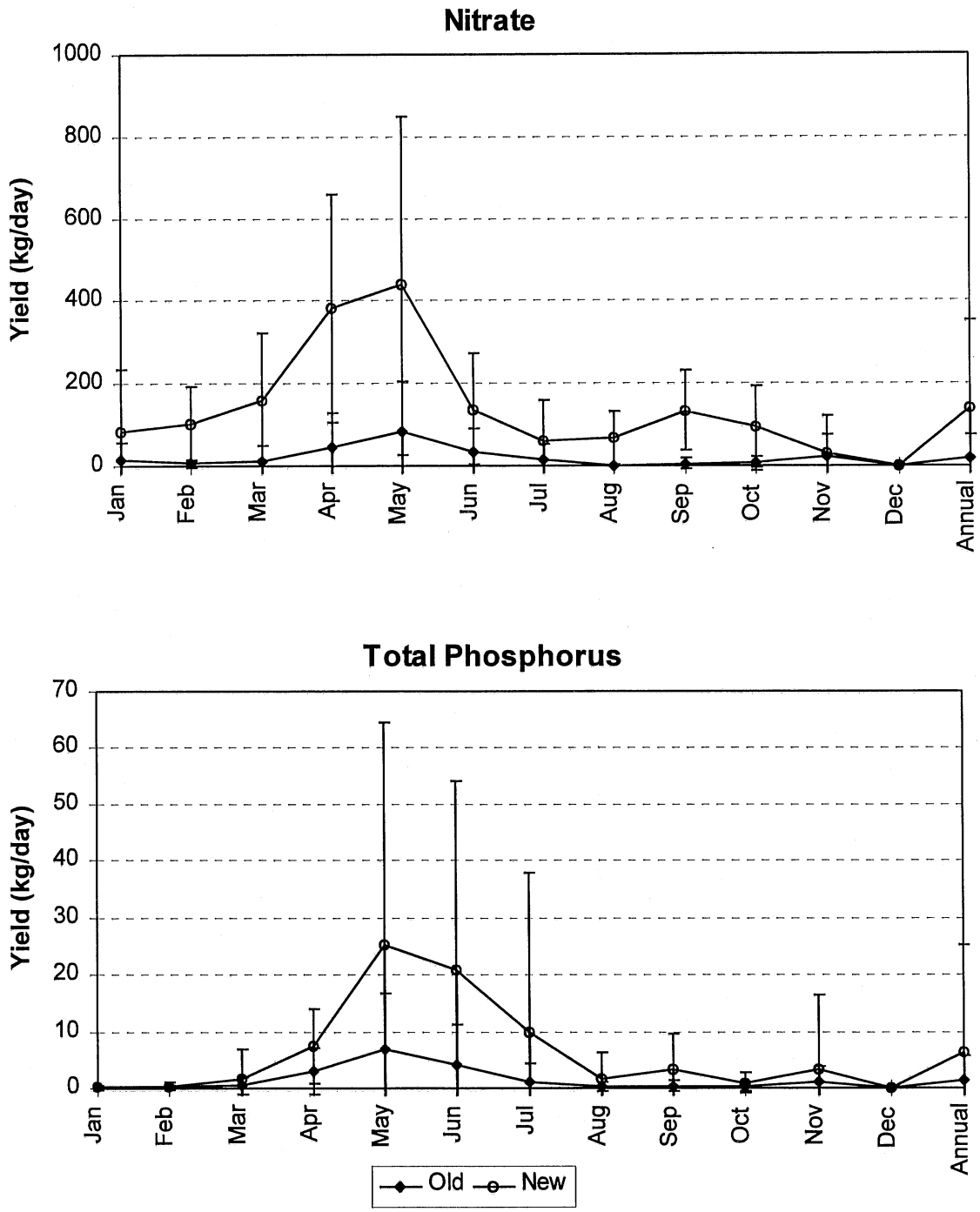


Figure 10 **(Cont'd.)** Comparison of simulated average monthly streamflow, nitrate yield, total phosphorus yield, sediment yield, and ammonia/organic N yield from the original SWAT96 (Old) and the modified version (New) to data measured in the Little Washita River near Ninnekah, OK, from 1963 to 1979 using the same (calibrated) input data for both versions.

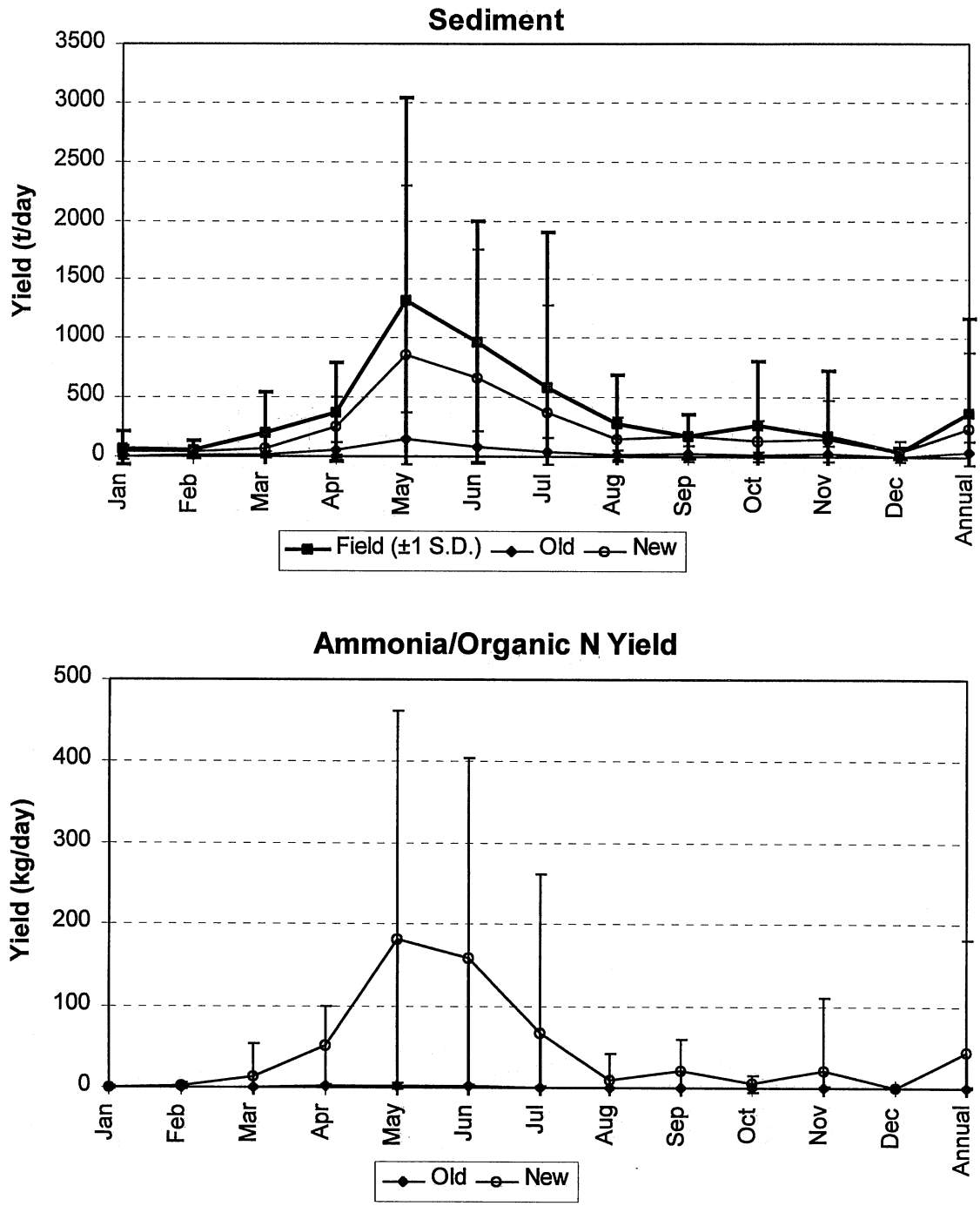


Figure 10 **(Cont'd.)** Comparison of simulated average monthly streamflow, nitrate yield, total phosphorus yield, sediment yield, and ammonia/organic N yield from the original SWAT96 (Old) and the modified version (New) to data measured in the Little Washita River near Ninnekah, OK, from 1963 to 1979 using the same (calibrated) input data for both versions.

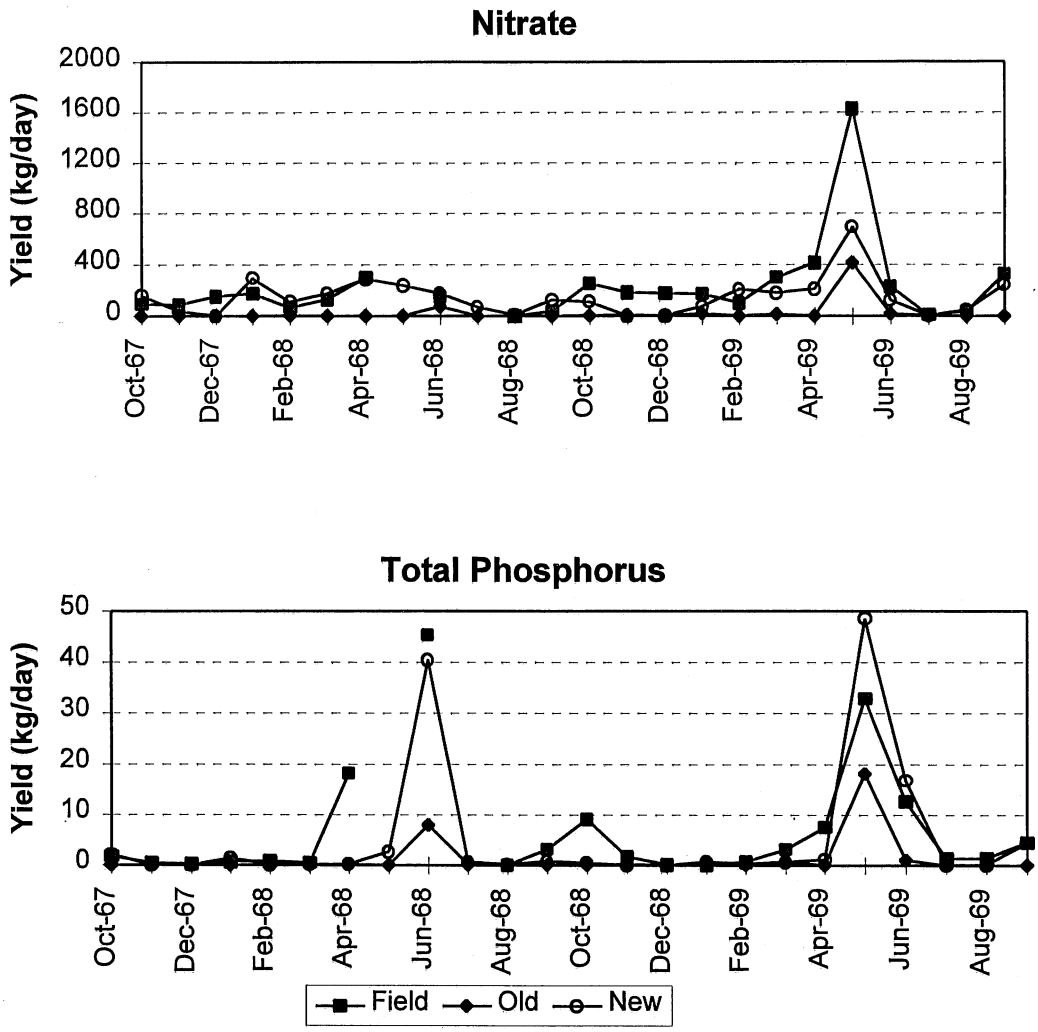


Figure 11 Comparison of simulated monthly nitrate yield, total phosphorus yield, and ammonia/organic N yield from the original SWAT96 (Old) and the modified version (New) to data measured in the Little Washita River near Ninnekah, OK, from October 1967 to September 1969 using the same (calibrated) input data for both versions.

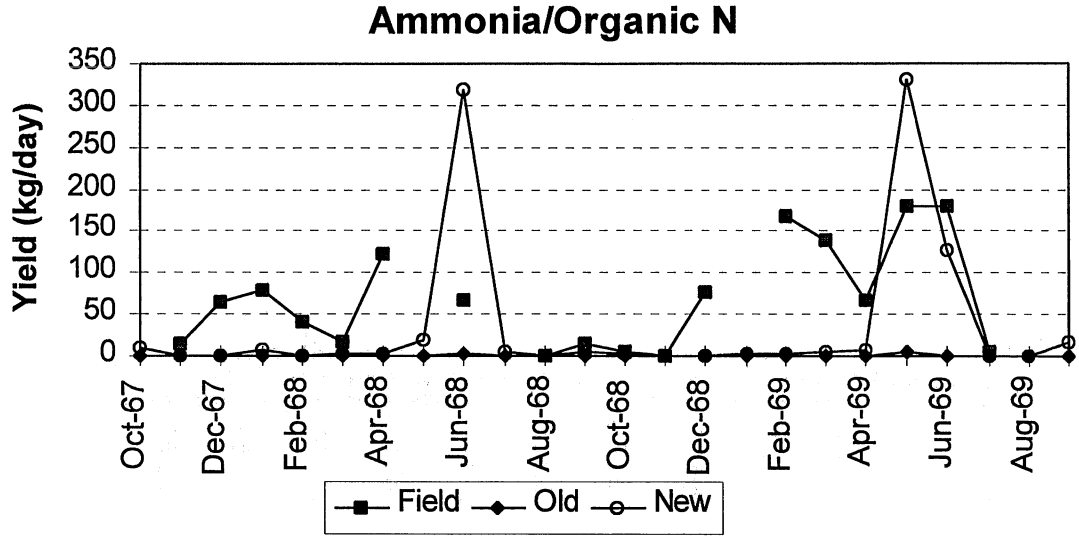


Figure 11 (Cont'd.) Comparison of simulated monthly nitrate yield, total phosphorus yield, and ammonia/organic N yield from the original SWAT96 (Old) and the modified version (New) to data measured in the Little Washita River near Ninnekah, OK, from October 1967 to September 1969 using the same (calibrated) input data for both versions.

VI. DISCUSSION/CONCLUSION

SWAT's crop growth submodel currently assumes that all crops are harvested by the end of the calendar year; the parameters for crop management operations are reset at the beginning of the year. In the southern U.S., however, there are often crops, such as winter wheat, that are planted in the fall and not harvested until January or February. It may be beneficial, then, to change SWAT so that a growing crop can be simulated beyond the first of the year. It is possible that the simulated streamflow in the Little Washita River during the months of September and October could be improved if such a change were made to the model.

SWAT was originally designed for agricultural watersheds and had been tested only in watersheds south of 43°N latitude (Arnold *et al.*, 1994; Arnold and Allen, 1996). The modifications made to SWAT extended the model's applicability to northern climates where snow accumulation and melt processes have a major role in watershed hydrology. The changes allow SWAT to be used to simulate forested watersheds as well. Therefore, if water quality and quantity data are available for calibration, SWAT can now be applied to any non-mountainous watershed in the U.S.

VII. REFERENCES

Anderson, E.A. 1967. A point energy and mass balance model of snow cover. U.S. Department of Commerce, National Oceanic and Atmospheric Administration, National Weather Service, *NOAA Technical Reports NWS 19.*

Arnold, J.G. and P.M. Allen. 1996. Estimating hydrologic budgets for three Illinois watersheds. *J. Hydrol.* **176**:57-77.

Arnold, J.G., Allen, P.M., and Bernhardt, G. 1993. A comprehensive surface-groundwater flow model. *J. Hydrol.* **142**:47-69.

Arnold, J.G., Williams, J.R., Srinivasan, R., King, K.W., and Griggs, R.H. 1994. SWAT - Soil and Water Assessment Tool - User Manual. USDA, Agricultural Research Service, Grassland, Soil and Water Research Laboratory, 808 East Blackland Road, Temple, TX 76502, 300 pp.

Arnold, J.G., Williams, J.R., Nicks, A.D., and Sammons, N.B. 1990. *SWRRB: A Basin Scale Simulation Model for Soil and Water Resources Management.* Texas A&M University Press, College Station, Texas, 204 pp.

Hanratty, M.P. and H.G. Stefan. 1997a. Using the IDRISI GIS to determine watershed model inputs. Submitted to *ASCE Hydrologic Engineering.*

Hanratty, M.P. and H.G. Stefan. 1997b. Simulating the effect of climate change on the hydrology and water quality of a northern forested watershed. Submitted to *Climatic Change.*

Hanratty, M.P. and H.G. Stefan. 1997c. Simulating the effect of climate change on the hydrology and water quality of a southern agricultural watershed. Submitted to *Agricultural Water Management.*

Hanratty, M.P. 1995. Modeling Continuous Non-point Source Inputs from Rural Watersheds to Lakes and Impoundments: A Review of Tools for Predicting Climate Change Effects. University of Minnesota St. Anthony Falls Laboratory Project Report 380, 47 pp.

Johanson, R.C., Imhoff, J.C., Kittle, J.L., and Donigian, A.S. 1984. Hydrologic Simulation Program-FORTRAN (HSPF): Users Manual for Release 8.0. EPA 600/3-84-066, U.S. Environmental Protection Agency, Athens, Georgia, 678 pp.

Kim, K., Chu, C.S., Bowers, C.E., and Baker, D.G.: 1974. Forecasting Rainfall and Snowmelt Floods on Upper Midwest Watersheds. University of Minnesota, St. Anthony Falls Laboratory. Project Report No. 151.

Koerper, G.J and Richardson, C.J.: 1980. Biomass and net annual primary production regressions for *Populus grandidentata* on three sites in northern lower Michigan. *Can. J. For. Res.* **10**:92-101.

Miles, P.D., Chen, C.M., and Leatherberry, E.C.: 1995. *Minnesota Forest Statistics, 1990, Revised.* U.S.D.A. Forest Service North Central Forest Experiment Station Resource Bulletin NC-158.

Osborn, H.B., Lane, L.J., Richardson, C.W., and Molnau, M.P. 1982. Precipitation. In Haan, C.T., Johnson, H.P., and Bakensiek, D.L. (eds.). *Hydrologic Modeling of Small Watersheds.* American Society of Agricultural Engineers, St. Joseph, Michigan, pp. 81-120.

Rango, A. and Martinec, J.: 1995. Revisiting the degree-day method for snowmelt computations. *Water Res. Bull.* **31**:657-669.

Schmitt, M.D.C. and Grigal, D.F.: 1981. Generalized biomass estimation equations for *Betula papyrifera* Marsh. *Can. J. For. Res.* **11:**837-840.

Srinivasan, R. and Arnold, J.G. 1994. Integration of a basin-scale water quality model with GIS. *Water Res. Bull.* **30**:453-462.

Wilson, W.T. 1941. An outline of the thermodynamics of snowmelt. *Am. Geophys. Union Trans., Part 1*:182-195.

Young, R.A., Onstad, C.A., Bosch, D.D. and Anderson, W.P. 1987. *AGNPS, Agricultural Non-Point-Source Pollution Model. A Watershed Analysis Tool.* U.S. Department of Agriculture, Morris, MN, Conservation Research Report 35, 80 pp.

Appendix

FORTRAN Code Changes

Appendix

addh.f

No changes

afert.f

Debugging

```
71,72c71,84
<         tfp = ano3 / 7.
<         ap(2,j) = ap(2,j) + tfp
---
> c         write (*,*) 'Auto P application, j, ida, iyr=', j,ida,iyr
> c Phosphorus added twice! commented out below and changed here so that
> c the fraction P in the fertilizer applied is used and so that it is
> c added to the layer specified by the user (lafert).
> c       tfp = ano3 / 7.
> c Assumes that fertilizer applied contists of both N and P and that
> c phosphorus stress won't occur without nitrogen stress.
>         if (afminn(j) .gt. 0) then
>             tfp = ano3 / afminn(j) * afminp(j)
>       else
>           tfp = 0.
>       endif
> c        ap(2,j) = ap(2,j) + tfp
>         ap(1,j) = ap(1,j) + tfp
86c98
<         ap(1,j) = ap(1,j) + dwfert * afminp(j)
---
> c        ap(1,j) = ap(1,j) + dwfert * afminp(j)
```

ahu.f

No changes.

alph.f

No changes.

apply.f

No changes

ascrv.f

No changes.

atri.f

No changes.

aunif.f

No changes.

blockd.f

```
Debugging
49c49
<       data spa /mpb * 0./
---
>       data spa /mpb * 0./, wflu /mbb * 0./
62,63c62,63
<       data prog /'  SW', 'AT  ', 'Dec.', '`95 ', 'SUNS', 'PARC', '
VER',
<      *    'SION', '95.2', 11 * '    '/
---
>       data prog /'  SW', 'AT  ', 'Mar.', '`97 ', 'SUNS', 'PARC', '
VER',
>      *    'SION', '96.2', 11 * '    '/
110c110
<      *nafert/mb*1/,nair/mb*1/, ngr /mb*1/,wflu/mbb*0./
---
>      *nafert/mb*1/,nair/mb*1/, ngr /mb*1/
```

clgen.f

First and third change to test sensitivity to omega_h, the effect of rain
on the relative humidity.
Second change part of tree growth model.

```
83c83
<       data l /1/, idist /1/
---
>       data l /1/, idist /1/, omega_h /0.9/
96c96,98
<
---
> c Mike Hanratty 7/23/97.  Reset leaf loss flag in spring
>       if (ida.lt.151 .and. dayl-1.ge.daylmn(j)) leaf(j) = 1
>
165c167
<         yy = 0.9 * wft(mo,j)
---
>         yy = omega_h * wft(mo,j)
168c170
<         if (precip.gt.0.0) rhm = rhm * 0.1 + 0.9
---
>         if (precip.gt.0.0) rhm = rhm * 0.1 + omega_h
```

clicon.f

Changed some of the climate change increments to ratios (multiplicative),
added increment for wind speed.
Second change is for debugging.

```
42,45c42,46
< c     rfinc(k,mo) |percent        |monthly adjustment in rainfall
< c     tmpinc(k,mo)|percent        |monthly adjustment in temperature
< c     huminc(k,mo)|percent        |monthly adjustment in relative
humidity
< c     radinc(k,mo)|percent        |monthly adjustment in solar
radiation
---
> c     rfinc(k,mo) |ratio          |monthly adjustment in rainfall
```

```
> c     tmpinc(k,mo)|deg C          |monthly adjustment in temperature
> c     huminc(k,mo)|fraction       |monthly adjustment in relative
humidity
> c     radinc(k,mo)|ratio          |monthly adjustment in solar
radiation
> c     uavminc(k,mo)|ratio         |monthly adjustment in wind speed
196,197c197,198
<           tmx(l) = tmxg + (vobmx(mo,l)-obmx(mo,j))
<           tmn(l) = tmng + (vobmn(mo,l)-obmn(mo,j))
---
>           tmx(l) = tmxg + (vobmx(mo,l)-obmx(mo,l))
>           tmn(l) = tmng + (vobmn(mo,l)-obmn(mo,l))
215c216
<         subp(k) = subp(k) * (1.+rfinc(k,mo)/100.)
---
>         subp(k) = subp(k) * rfinc(k,mo)
219c220
<         ra(k) = ra(k) + radinc(k,mo)
---
>         ra(k) = ra(k) * radinc(k,mo)
220a222
>       u10(j) = u10(j) * uavminc(k,mo)
```

coefs.f

No changes.

comand.f

Change of subroutine name.

```
52c52
<    90       call transfer(icode,ihout,inum1,inum2,inum3,rnum1,inum4)
---
>    90       call transferw(icode,ihout,inum1,inum2,inum3,rnum1,inum4)
```

common.f

Debugging

```
12c12
< c     mcr = max number of crops per year
---
> c     mcr = max number of crops per year (plus 1)
```

Increased the capacity of the code to more subbasins.

```
22,24c22,24
<       parameter (mb = 10)
<       parameter (mch = 10)
<       parameter (mbb = 2)
---
>       parameter (mb = 500)
>       parameter (mch = 500)
>       parameter (mbb = 100)
```

Debugging

```
30c30
<       parameter (mcr = 3)
---
>       parameter (mcr = 4)
```

< Original version
> New version

```
Test of different runoff algorithm.
58c58
<      *    skoc(mpdb), nreach, iprp, ipdwql, abco, ei, usle
---
>      *    skoc(mpdb), nreach, iprp, ipdwql, abco, ei, usle,
sumcnfc(mb)

Addition of wind speed increment.
70c70
<      *    tnylda(mnr,mcr,mb), fertef(mb), cnyld(mcrdb),
---
>      *    tnylda(mnr,mcr,mb), fertef(mb), cnyld(mcrdb),
uavminc(mb,12),

Addition of snow evaporation constant.
116c116
<      *    nsim, msim, tmx(mb), tmn(mb), tb(mcrdb), to(mcrdb),
---
>      *    nsim, msim, tmx(mb), tmn(mb), tb(mcrdb), to(mcrdb),
snoevfac,

Debugging
120c120
<      *    chtmx(mcrdb), cht(mb), dewpt(12,mb), uavm(12,mb), co2(mb),
---
>      *    chtmx(mcrdb), cht(mb), dewpt(12,mb), uavm(12,mb),
co2(12,mb),

Tree growth submodel.
133c133,134
<      *    ihvo(mnr,mcut,mb), phun(mnr,ma,mb), bn(4,mcrdb), anano3(mb)
---
>      *    ihvo(mnr,mcut,mb), phun(mnr,ma,mb), bn(4,mcrdb), anano3(mb),
>     *    leaf(mb)

Snow melt model
139c140,141
<      *    fda(10,mb), elevb(10,mb), snoeb(10,mb), tlaps(mb)
---
>      *    fda(10,mb), elevb(10,mb), snoeb(10,mb), tlaps(mb),
snomx(mb),
>     *    pmelt(mb)

Debugging.
212c214,215
<      *    rodir(mstep,mb), dt, ievent,sv,cbodu,dox,chl_a
---
>      *    rodir(mstep,mb), dt, ievent
>     common /missed/ dox, cbodu, chl_a, tot, sv


crpmd.f

No changes.

curno.f

Debugging
12c12
< c      sumfc(j)    |              |root zone water content at field
capacity
```

```
---
> c      sumfc(j)    |              |water content at field capacity
```

decay.f

No changes.

delr.f

No changes.

dstg.f

No changes.

dstn1.f

No changes.

ee.f

No changes.

eiusle.f

No changes.

enrrt.f

No changes.

enrsb.f

No changes.

erfc.f

No changes.

etact.f

Snow evaporation, debugging.

```
65,70c65,72
< c     if (sno(j).ge.eos) then
< c       take all soil evap from snow cover
< c       sno(j) = sno(j) - eos
< c       es = eos
< c       snoev = snoev + eos * flu(j)
< c     else
---
>     if (elevb(1,j).le.0.) then
> c Not using elevation bands for snow evaporation
>         if (sno(j).ge.eos) then
> c         take all soil evap from snow cover
>           sno(j) = sno(j) - eos
>           es = eos
>           snoev = snoev + eos * flu(j)
>         else
72,75c74,78
< c       esleft = esleft - sno(j)
```

< Original version
> New version

```
< c       snoev = snoev + sno(j) * flu(j)
< c       sno(j) = 0.
<
---
>           esleft = esleft - sno(j)
>           snoev = snoev + sno(j) * flu(j)
>           sno(j) = 0.
>         endif
>       else
77,86c80,90
<         snoevb = 0.
<         do 77 ib = 1, 10
<           if (fda(ib,j).le.0.) go to 78
<           if (snoeb(ib,j).gt.0.) then
<             if (snoeb(ib,j).lt.es) then
<               snoevb = snoevb + snoeb(ib,j) * fda(ib,j)
<               snoeb(ib,j) = 0.
<             else
<               snoeb(ib,j) = snoeb(ib,j) - es
<               snoevb = snoevb + es * fda(ib,j)
---
>         snoevb = 0.
>         do 77 ib = 1, 10
>           if (fda(ib,j).le.0.) go to 77
>           if (snoeb(ib,j).gt.0.) then
>             if (snoeb(ib,j).lt.es) then
>               snoevb = snoevb + snoeb(ib,j) * fda(ib,j)
>               snoeb(ib,j) = 0.
>             else
>               snoeb(ib,j) = snoeb(ib,j) - es
>               snoevb = snoevb + es * fda(ib,j)
>             endif
88,91c92,96
<           endif
<     77 continue
<     78 esleft = esleft - snoevb
<        snoev = snoevb * flu(j)
---
>     77    continue
>     78    esleft = esleft - snoevb
>           snoev = snoevb * flu(j)
>         sno(j) = max(sno(j) - snoevb, 0.)
>       endif


etpot.f

Debugging.
30c30
< c     co2(j)       |ppm           |co2 concentration
---
> c     co2(mo,j)    |ppm           |co2 concentration for each month

Snow evaporation
86a87
>       eaj = eaj*snoevfac

Debugging
178,179c179,180
< c           rc = 2.0 / ((alai(j)+0.01)*g1*(1.4-0.4*co2(j)/330.0))
<             rc = 2.0 / ((alai(j)+0.01)*g1*exp(.00155*(330.-co2(j))))
```

< Original version
> New version

```
---
> c             rc = 2.0 / ((alai(j)+0.01)*g1*(1.4-0.4*co2(mo,j)/330.0))
>            rc = 2.0 / ((alai(j)+0.01)*g1*exp(.00155*(330.-
co2(mo,j))))
```

expo.f

No changes.

fert.f

No changes.

finalbal.f

```
Debugging
133c133
<         if (ires.eq.1) xz = 1.
---
> c        if (ires.eq.1) xz = 1.
```

gcycl.f

No changes.

graze.f

No changes.

grow.f

```
Debugging
23c23
< c     co2(j)       |(ppm)          |co2 concentration
---
> c     co2(mo,j)    |(ppm)          |co2 concentration for each month
120,121c120,121
<           beadj = 100. * co2(j) / (co2(j)+exp(
<      *         wac21(ncr(nro(j),icr(j),j))-co2(j)*
---
>           beadj = 100. * co2(mo,j) / (co2(mo,j)+exp(
>      *         wac21(ncr(nro(j),icr(j),j))-co2(mo,j)*
```

gwmod.f

No changes.

icl.f

No changes.

irrigate.f

No changes.

jdt.f

No changes.

< Original version
> New version

lakeq.f

No changes.

lakes.f

No changes.

main.f

formatting, units change

```
212a213,214
>     write (80,5410) iyr
> 5410  format (i4,' is the first year')
223a226,227
>       write (80,*) "END"
>     close (80)
225c229
<  5000 format (//6x,'SEG GIS BIG MGT MON ',' AREA KM2 ',' PREC MM ',
---
>  5000 format (//15x,'SEG GIS BIG MGT MON ',' AREA KM2 ',' PREC MM ',
238c242
<      *    '  MFRSP KH','     SW MM  ',' BIOM KH ','   LAI KH''   YLD
KH'
---
>      *    '  MFRSP KH','     SW MM  ',' BIOM TH ','   LAI KH''   YLD
KH'
265,272c269,285
<  5400 format (//7x,'RCH BGIS DAY',t21,'WTEMP (C)',t33,'ALGIN ppm',t45,
<      *    'ALGOUT ppm',t57,'ORGNIN ppm',t69,'ORGNOUT ppm',t81,
<      *    'AMMIN ppm',t93,'AMMOUT ppm',t105,'NO2in ppm',t117,
<      *    'NO2out ppm',t129,'NO3IN ppm',t141,'NO3OUT ppm',t153,
<      *    'ORGPIN ppm',t165,'ORGPOUT ppm',t177,'DISPIN ppm',t189,
<      *    'DISPOUT ppm',t201,'CBODIN ppm',t213,'CBODOUT ppm',t225,
<      *    'SOXY ppm',t237,'DISOXIN ppm',t249,'DISOXOUT ppm',t261,
<      *    'FLOW m^3')
---
> c Changed to kg/day from mg/L (or ppm) 6/17/97 Mike H.
>  5400 format (//7x,'RCH BGIS DAY',t21,'WTEMP (C)',t33,'ALGIN kgd',t45,
>      *    'ALGOUT kgd',t57,'ORGNIN kgd',t69,'ORGNOUT kgd',t81,
>      *    'AMMIN kgd',t93,'AMMOUT kgd',t105,'NO2in kgd',t117,
>      *    'NO2out kgd',t129,'NO3IN kgd',t141,'NO3OUT kgd',t153,
>      *    'ORGPIN kgd',t165,'ORGPOUT kgd',t177,'DISPIN kgd',t189,
>      *    'DISPOUT kgd',t201,'CBODIN kgd',t213,'CBODOUT kgd',t225,
>      *    'SOXY kgd',t237,'DISOXIN kgd',t249,'DISOXOUT kgd',t262,
>      *    'FLOW mmd',t271, 'SUS.SED t/d',t283'TRAVEL TIME hrs')
> c 5400 format (//7x,'RCH BGIS DAY',t21,'WTEMP (C)',t33,'ALGIN
ppm',t45,
> c     *    'ALGOUT ppm',t57,'ORGNIN ppm',t69,'ORGNOUT ppm',t81,
> c     *    'AMMIN ppm',t93,'AMMOUT ppm',t105,'NO2in ppm',t117,
> c     *    'NO2out ppm',t129,'NO3IN ppm',t141,'NO3OUT ppm',t153,
> c     *    'ORGPIN ppm',t165,'ORGPOUT ppm',t177,'DISPIN ppm',t189,
> c     *    'DISPOUT ppm',t201,'CBODIN ppm',t213,'CBODOUT ppm',t225,
> c     *    'SOXY ppm',t237,'DISOXIN ppm',t249,'DISOXOUT ppm',t261,
> c     *    'FLOW m^3')
```

nfix.f

< Original version
> New version

```
No change.

nlch.f

No change.

nminrl.f

No changes.

npup.f

Tree growth submodel
41c41,42
< C       GET 'EM WHERE YOU CAN BOYS
---
> C       GET 'EM WHERE YOU CAN BOYS unless you're a tree, then skip
first layer
> C                               Mike Hanratty 12/9/96
42a44
>          if(idc(ncr(nro(j),icr(j),j)).ne.7 .or. m.ne.1) then
50a53
>          end if

nrain.f

Debugging
23c23,24
<       wno3(1,j) = wno3(1,j) + .0008 * precip
---
> c Factor on precip was 0.0008, should be 0.008. - Mike H. 9/3/97
>       wno3(1,j) = wno3(1,j) + .008 * precip

nup.f

Tree growth
44c44,45
< C       GET 'EM WHERE YOU CAN BOYS
---
> C       GET 'EM WHERE YOU CAN BOYS unless you're a tree, then skip
first layer
> C                               Mike Hanratty 12/9/96
45a47
>          if(idc(ncr(nro(j),icr(j),j)).ne.7 .or. m.ne.1) then
53a56
>          end if

nuts.f

No changes.

open.f

No changes.

open2.f

No changes.

open3.f
```

< Original version
> New version

```
No changes.

operatn.f

Debugging
19c19
< c     day1
---
> c     dayl

Debugging
64a65
> c     kill              logical        true if crop has been killed
during current day
69a71,73
>       logical kill
>
>       kill = .false.

Tree growth
97a102
> c Whole trees are not perennial, left out of kill. Mike H. 7/23/97
100c105
<      *    ida.gt.350) then
---
>      *    idc(ncr(nro(j),icr(j),j)).ne.7.and.ida.gt.350) then
119a125
>         kill = .true.
159c165,166
<    40   if (ihvo(nro(j),ncut(j),j).gt.0) then
---
>    40   if (ihvo(nro(j),ncut(j),j).gt.0.and.
>     *          leaf(j).gt.0) then
182c189,205
<           alai(j) = alai(j) * dm(j) / xx
---
> c Mike Hanratty 7/23/97
> c If ncr = 26 or 32, crop is mixed or deciduous forest, respectively,
and
> c this represents the loss of leaves in the fall, all yield (leaf
loss)
> c goes to residue and the leaf area index is cut by 90% for a mixed
forest
> c and set to zero for a deciduous forest.
>           if (ncr(nro(j),icr(j),j).eq.26 .or.
>     .          ncr(nro(j),icr(j),j).eq.32) then
>             rsd(j) = rsd(j) + yield
>             leaf(j) = 0
>             if (ncr(nro(j),icr(j),j).eq.26) then
>                   alai(j) = 0.1*alai(j)
>             else
>                   alai(j) = 0.
>             endif
>         else
>             alai(j) = alai(j)*dm(j)/xx
>         endif

Debugging
197c220
<           dmanu(j) = dmanu(j) + yield / 1000.
---
```

< Original version
> New version

```
>           dmanu(j) = dmanu(j) + dm(j) / 1000.
207a231
>         kill = .true.
226c250,254
<         if (g(j).gt.phut(nro(j),nop(j),j)) go to 90
---
>         if (igro(j).eq.1.or.kill) then
>          if (g(j).gt.phut(nro(j),nop(j),j)) go to 90
>         else
>          if (gplt(j).gt.phut(nro(j),nop(j),j)) go to 90
>       endif
```

orgn.f

```
Very important debugging
37c37,38
<       yon = .001 * conn * yd / da9
---
> c Corrected units conversion. Greatly improved org N simulations! -
Mike H. 8/7/97
>       yon = .001 * conn * yd / (da9*flu(j))
```

outpest.f

No changes.

outsoil.f

No changes.

outsub.f

No changes.

perc.f

```
Debugging.
86c87
<       if (prk.lt.sup) prk = 0.
---
> c      if (prk.lt.sup) prk = 0.
```

pestlch.f

No changes.

pestq.f

No changes.

pestrisk.f

No changes.

pestw.f

No changes.

pesty.f

< Original version
> New version

No changes.

pkq.f

No changes.

pkrn.f

```
Testing percolation algorithms
28,29c51,52
<         xx = 0.002 * sc(j1,j)
<         cf(j) = xx
---
>         xx = 0.003 * sc(j1,j)
>         cf(j) = xx
```

pminrl.f

No changes.

pond.f

All debugging

```
16c16
< c    qd             |                |pond inflow
---
> c    qd             |                |streamflow
40,50c40,49
< c    sa             |
< c    precip         |(mm)            |precipitation
< c    afp            |
< c    q1
< c    ev
< c    sp             |                |pond seepage
< c    rl             |                |pond rainfall
< c    v(j)           |(ha-m)          |initial pond volumes
< c    vv             |(ha-m)          |initial pond volumes
< c    o
< c    cs(j)          |(ppm)           |initial sediment concentration in
ponds
---
> c    sa             |(ha)            |pond surface area
> c    q1                 |(m^3)           |pond inflow
> c    ev                 |(m^3)                 |pond evaporation
> c    sp             |(m^3)           |pond seepage
> c    rl             |(m^3)           |pond rainfall
> c    v(j)           |(m^3)           |pond volume
> c    vv             |(m^3)           |initial pond volumes
> c    o                  |(m^3)                 |pond outflow
> c    cs(j)          |(ppm)           |sediment concentration in ponds
> c    twl            |(mm)            |pond storage (inflow - outflow)
67,69c66,67
<         q1 = qd
<         eo = pet(j)
<         precip = subp(j)
---
> c Units conversion changed for q1 - Mike H. 8/6/97
>         q1 = qd*da*flu(j)*10.
71,72c69
<         afp = aff * fp(j)
```

< Original version
> New version

```
<          q1 = qdr * 10. * (afp/10.-sa)
---
>          q1 = qdr * 10. * (da*flu(j)*fp(j)-sa)
74,76c71,77
<        ev = 7. * eo * sa
<        sp = hc(j) * sa * 240.
<        rl = precip * sa * 10.
---
> c Evaporation and precip do not differ between ponds that drain entire
> c watersheds and those that drain only a portion. - Mike H. 8/6/97
>        ev = 7. * pet(j) * sa
> c Include seepage through the dam. Mike H. 8/5/97
> c       sp = hc(j) * sa * 240.
>        sp = hc(j) * sa * 240. + sepp(j)
>        rl = subp(j) * sa * 10.
133,137c134,137
<          if (fp(j).ge.0.999) then
<            pq = qd - o
<          else
<            pq = (qdr*afp-o) / aff
<          end if
---
>
> c pq used for both pond storage and pond sediment inflow,  when
> c used as pond storage, changed to twl - Mike H. 8/6/97
>          twl = (q1-o)/(da*flu(j)*10.)
```

pondcon.f

```
Debugging
53c53,55
<          if (fp(j).lt.1.0.and.fp(j).gt.0.) then
---
> c pq used for both pond storage and pond sediment inflow,  changed
> c pond storage uses to twl - Mike H. 8/5/97
>          if (fp(j).lt.0.999.and.fp(j).gt.0.) then
59c61
<            ssb(33) = ssb(33) + pq * flu(j)
---
>            ssb(33) = ssb(33) + twl * flu(j)
73d74
<            twl = pq
```

psed.f

No changes.

purk.f

```
Returned crack flow
94c94
< c            call pkrn(j)
---
>              call pkrn(j)
```

qman.f

No changes.

randn.f

< Original version

> New version

No changes.

readbsn.f

```
Snow evaporation
55a56
>     read (11,5000) snoevfac
59a61
>     if (snoevfac.le.0.) snoevfac = 1.0
```

readchm.f

No changes.

readcrop.f

No changes.

readfert.f

No changes.

readfig.f

No changes.

readfile.f

No changes.

readgw.f

No changes.

readinpt.f

No changes.

readinst.f

No changes.

readlwq.f

No changes.

readmco.f

```
Debugging.
72,75c72,86
<       read (18,5300) titldum
< c     read (18,5200) ansf(i),fnmx(i),anmx(i),iafert(i), ipman(i),
< c    *    lafert(i), afminn(i), afminp(i), aforgn(i), aforgp(i),
< c    *    afnh3n(i), fertef(i), nstrs(i)
---
> c      read (18,5300) titldum
> c Automatic fertilization was commented out!  Mike H. 4/30/97
>       read (18,5200) ansf(i),fnmx(i),anmx(i),iafert(i), ipman(i),
>      *    lafert(i), afminn(i), afminp(i), aforgn(i), aforgp(i),
```

< Original version
> New version

```
>      *    afnh3n(i), fertef(i), nstrs(i)
>           if (fnmx(i).le.0.) fnmx(i) = 100.
>           if (anmx(i).le.0.) anmx(i) = 500.
>           if (fertef(i).le.0.) fertef(i) = 1.3
> c         default is commercial fert (28-10-10)
>           if (iafert(i).le.0) iafert(i) = 50
>         afminn(i) = fminn(iafert(i))
>           afminp(i) = fminp(iafert(i))
>           aforgn(i) = forgn(iafert(i))
>           aforgp(i) = forgp(iafert(i))
>           afnh3n(i) = fnh3n(iafert(i))
```

readmgt.f

Debugging

```
117a118,120
> c Mike Hanratty 7/23/97 When crop is growing at start of simulation,
> c set alai and g based on maximum leaf area index (blai).
>       if (alai(i).le.0.) alai(i) = 0.1*blai(ncrp)
120c123,124
<         g(i) = .1
---
>         g(i) = alai(i)/blai(ncrp)
> c        g(i) = .1
121a126,133
> c Mike Hanratty 7/23/97 When crop is growing at start of simulation,
> c calculate previous nutrient use.
>       if (dm(i).gt.0.) then
>           cnb = (bn(4,ncrp) - bn(3,ncrp)) * (1. - g(i) / (g(i) +
>     *        exp(bn(1,ncrp) - bn(2,ncrp) * g(i)))) + bn(3,ncrp)
>             snup(i) = cnb * dm(i)
>             spup(i) = snup(i)/7.1
>         endif
```

readpest.f

No changes.

readpnd.f

Debugging

```
59a60,64
> c Set dam seepage to 1 mm per day
>     if (sepp(i).eq.0.) sepp(i) = 10.*sax(i)
> c Set pond/wetland bottom hyd. conductivity to channel hydr.
conductivity
>       if (hc(i).eq.0.) hc(i) = chk(2,i)
>       if (hcw(i).eq.0.) hcw(i) = chk(2,i)
```

readpts.f

No changes.

readres.f

No changes.

readrte.f

```
Debugging
44,46c44,47
<       if (chw(2,i).lt.200.) chw(2,i) = 200.
<       if (chd(i).lt.4.) chd(i) = 4.
<       chxk(i) = .028
---
> c  The following three lines commented out by Mike H. 5/23/97
> c       if (chw(2,i).lt.200.) chw(2,i) = 200.
> c       if (chd(i).lt.4.) chd(i) = 4.
> c       chxk(i) = .028


readsol.f

No changes.

readsub.f

Change to climate change increments
33c33
< c     rfinc       |(%)           |monthly precipitation increment
---
> c     rfinc       |(ratio)       |monthly precipitation increment
35,36c35,37
< c     radinc      |(langleys)    |monthly solar radiation increment
< c     huminc      |(%)           |monthly relative humidity increment
---
> c     radinc      |(ratio)       |monthly solar radiation increment
> c     huminc      |(fraction)    |monthly relative humidity increment
> c     uavminc     |(ratio)       |monthly wind speed increment

Debugging
52c53
<       read (10,5000) flu(i), cn2(i), co2(i), sno(i), chl(1,i), chs(i),
---
>       read (10,5000) flu(i), cn2(i), co2(1,i), sno(i), chl(1,i),
chs(i),
54c55,58
<       if (co2(i).le.0.) co2(i) = 330.
---
>     if (co2(1,i).le.0.) co2(1,i) = 330.
>       do 5 mo=2,12
>       co2(mo,i) = co2(1,i)
> 5   continue
66c70
<       if (stp(i).gt.0.0506) ecp(i) = .0105 * stp(i) ** -1.5279
---
> c      if (stp(i).gt.0.0506) ecp(i) = .0105 * stp(i) ** (-1.5279)

Addition of wind speed increment.
74a79
>       read (10,5200) (uavminc(i,mo),mo = 1,12)

Debugging
81a87,88
> c Changed dimension of co2 so that the monthly increment would work.
> c Mike H. 7/10/97
83c90
<         co2(i) = co2(i) + co2inc(i,mo)
---
```

< Original version
> New version

```
>          co2(mo,i) = co2(mo,i) + co2inc(i,mo)
```

readtill.f

No changes.

readwgn.f

No changes.

readwql.f

No changes.

recday.f

No changes.

recepic.f

No changes.

recmon.f

No changes.

res.f

No changes.

resopmrb.f

No changes.

rflow.f

```
Attempt at modeling tile drainage (not used)
45a46,53
> c Insert variable to represent tile drainage - Mike H. 8/4/97
> c          if
(ncr(nro(j),icr(j),j).lt.26.and.ncr(nro(j),icr(j),j).ne.22
> c     *
      .and.ncr(nro(j),icr(j),j).ne.19.and.sc(ii,j).le.50.0.and.
> c       *          ii.eq.nn) then
> c                            tile = 1000.
> c         else
>                              tile = 1
> c         endif
48c56
<             sumk = sumk + dg / sc(ii,j)
---
>             sumk = sumk + dg / (sc(ii,j)*tile)
```

route.f

```
Debugging
101c101
< c     srch(2,*)    |              |outfloe
---
```

< Original version
> New version

```
> c     srch(2,*)    |              |outflow
173c173,174
<       shallst(j) = shallst(j) + tlc / (1000.*da*flu(j))
---
>     temp=af*flu(j)
>       shallst(j) = shallst(j) + tlc / temp
184a186,191
> c Added sediment routing back in - should be before flow reaches the
pond
> c Mike H. 6/18/97
>         if (inum1.ne.inum2 .and. qd.gt.0.) then
>          call rtsed(j,ihout,inum1,inum2,inum3,rnum1)
>         endif
>
212c219,221
<         twl = pq
---
> c pq used for both pond storage and pond sediment inflow, when used
for
> c pond storage, changed to twl - Mike H. 8/5/97
> c         twl = pq
215c224,227
<
---
> c Moved assignment of varoute(2,*) to here so that the flow is correct
> c in watqual for printout and made sure flow is greater than or equal
> c to zero. Mike H. 7/10/97
>       varoute(2,ihout) = max(qd,0.0)
244c256
<       varoute(2,ihout) = qd
---
> c      varoute(2,ihout) = qd
308c320
<       if (ipd.eq.1) then
---
>       if (ipd.eq.1.and.iyr.gt.nyskip) then

Eliminated some of the daily writes to save on disk space.
319,321c331,333
<             write (8,5000) ij, ix, ix, ida, xx, xx, xx, xx, xx, xx,
xx,
<      *          xx, xx, xx, xx, xx, xx, xx, xx, xx, xx, xx, xx, xx,
xx,
<      *          xx, xx, xx, xx, xx, xx, xx, xx
---
> c            write (8,5000) ij, ix, ix, ida, xx, xx, xx, xx, xx, xx,
xx,
> c     *          xx, xx, xx, xx, xx, xx, xx, xx, xx, xx, xx, xx, xx,
xx,
> c     *          xx, xx, xx, xx, xx, xx, xx, xx
325,334c337,346
<         write (8,5000) j, nbgis(j), nbigs(j), ida, dart(j),
<      *      varoute(2,inum2) / 86400., qd / 86400., varoute(3,inum2),
<      *      yd, varoute(3,inum2) - yd, varoute(4,inum2) * dart(ihout)
*
<      *      100., yon * dart(ihout) * 100., varoute(5,inum2) *
<      *      dart(ihout) * 100., yph * dart(ihout) * 100., evp *
<      *      dart(ihout) / 86400., tlc * dart(ihout) / 86400., rl *
<      *      dart(ihout) / 86400., varoute(6,inum2) * dart(ihout) *
100.,
<      *      yno3 * dart(ihout) * 100., varoute(7,inum2) * 100., yph *
```

< Original version
> New version

```
<       *      dart(ihout) * 100., solpesti, solpesto, sorpesti,
sorpesto,
<       *      react, volat, setl, resus, -difus, reactb, bury, cpest2(j)
---
> c        write (8,5000) j, nbgis(j), nbigs(j), ida, dart(j),
> c     *      varoute(2,inum2) / 86400., qd / 86400., varoute(3,inum2),
> c     *      yd, varoute(3,inum2) - yd, varoute(4,inum2) * dart(ihout)
*
> c     *      100., yon * dart(ihout) * 100., varoute(5,inum2) *
> c     *      dart(ihout) * 100., yph * dart(ihout) * 100., evp *
> c     *      dart(ihout) / 86400., tlc * dart(ihout) / 86400., rl *
> c     *      dart(ihout) / 86400., varoute(6,inum2) * dart(ihout) *
100.,
> c     *      yno3 * dart(ihout) * 100., varoute(7,inum2) * 100., yph *
> c     *      dart(ihout) * 100., solpesti, solpesto, sorpesti,
sorpesto,
> c     *      react, volat, setl, resus, -difus, reactb, bury,
cpest2(j)


routres.f

No changes.

routsub.f

No changes.

rteinit.f

No changes.

rthyd.f

Debugging
63c63
<       if (sdti.gt.0.) then
---
>       if (sdti.gt.1.e-30) then
71c71,74
<         scoef = 1.
---
> c This line commented out to restore the storage coefficient.  All
> c flow was routed through the subbasin in a day - not accurate for
> c larger watersheds. Mike H. 7/24/97
> c        scoef = 1.


rtorgn.f

No changes.

rtpest.f

No changes.

rtpsed.f

No changes.

rtsed.f
```

< Original version
> New version

```
Debugging
87,119d86
< c     dur = 24.0
< c     trt = chl(2,j) / (3.6*vc)
< C     CALCULATE DEPOSITION
< c     sum = 0.
< c     do 10 m = 1, nsz
< c       yi = vs(m) * trt
< c       if (yi.ge.d) then
< c         dr = 0.5 * d / yi
< c       else
< c         dr = 1. - 0.5 * yi / d
< c       end if
< c       sum = sum + pct(m,j) * dr
< c  10 continue
< c     CALCULATE DEGRADATION
< c     dep = yd * (1.-sum)
< c     dur = dur * 3600.
< c     tpk = tc(j) * 3600.
< c     b = .5 * tpk * pr / qdin
< c     if (b.gt..375) b = .375
< cc RAMA
< c     prf = .5 * b + .234
< c     dgr = alc(j) * dur * chw(2,j) * (prf*d*chss(j)*vc) ** spexp
< c     dgb = 0.
< c     if (dgr.gt.dep) then
< c       dd = dgr - dep
< c       dgb = chxk(j) * chc(j) * dd
< c       dgr = dep
< c     end if
< c     yd = yd + (dgr+dgb-dep)
< c     dep = yd * (1.-sum)
< c     yd = yd - dep
< c     return
< c     end


save.f

No changes.

simulate.f

Debugging
171a172
>       mo1 = mo

Added an annual write to the screen to track progress.
172a174
>       write (*,*) ' Beginning daily loop, year = ', iyr

Debugging
216a223
> c           determine month for next day
222,223c229,235
< c           determine month for next day
<             call writem(j)
---
> c writem and writea calls moved here inside if-then block so that the
```

```
> c subroutines are only called when necessary - more efficient -- Mike
H. 4/15/97
>              if (mo.ne.mo1.and.ipd.ne.1) then
>                   call writem(j)
>                     if (mo.lt.mo1) call writea(j)
>                 mo1 = mo
>              endif


snom.f

Snowmelt algorithm
3,4c3,4
< c     this subroutine predicts daily snom melt when the average air
< c     temperature exceeds 0 degrees centigrade
---
> c     this subroutine predicts daily snom melt when the maximum (or
> c     average) air temperature exceeds 0 degrees centigrade
33,37c33,37
<         do 1 ib = 1, 10
<           if (fda(ib,j).le.0.) go to 2
<           tdif = (elev(j) - elevb(ib,j)) * tlaps(j) / 1000.
<           txeb = tx(j) + tdif
<           tmxeb = tmx(j) + tdif
---
>           do 1 ib = 1, 10
>             if (fda(ib,j).le.0.) go to 1
>             tdif = (elev(j) - elevb(ib,j)) * tlaps(j) / 1000.
>             txeb = tx(j) + tdif
>             tmxeb = tmx(j) + tdif
39c39,40
<           if (txeb.lt.0.) then
---
>             if (txeb.lt.0.) then
> c            if (tmxeb.lt.0.) then
41,43c42,44
<             snoeb(ib,j) = snoeb(ib,j) + precip
<             snow = snow + precip * fda(ib,j)
<           else
---
>               snoeb(ib,j) = snoeb(ib,j) + precip
>               snow = snow + precip * fda(ib,j)
>             else
45,50c46,60
<             smleb = 4.57 * tmxeb
<             if (smleb.gt.snoeb(ib,j)) smleb = snoeb(ib,j)
<             snoeb(ib,j) = snoeb(ib,j) - smleb
<             sml = sml + smleb * fda(ib,j)
<           endif
<       1 continue
---
> c             smleb = 4.57 * tmxeb
>               if (pmelt(j).le.10) then
>                a = 0.457
>             elseif (pmelt(j).ge.35) then
>                  a = 3.66
>               else
>                  a = 0.00549*pmelt(j)*pmelt(j) - 0.11887*pmelt(j) +
1.0973
>               endif
> c           smleb = a * tmxeb
```


< Original version
> New version

```
>            smleb = a * txeb
>              if (smleb.gt.snoeb(ib,j)) smleb = snoeb(ib,j)
>              snoeb(ib,j) = snoeb(ib,j) - smleb
>              sml = sml + smleb * fda(ib,j)
>            endif
>     1    continue
52,56d61
< c     add/sub aggregate snow fall and melt from effective precip
< c     and snow cover
<
<     2 precip = precip + sml -snow
<       sno(j) = sno(j) + snow - sml
60,66c65,85
<       if (tmx(j).gt.0.) then
<         sml = 4.57 * tmx(j)
<         if (sml.gt.sno(j)) sml = sno(j)
<         sno(j) = sno(j) - sml
<         precip = precip + sml
<       else
<         sml = 0.
---
> c        if (tmx(j).gt.0.) then
>         if (tx(j).gt.0.) then
> c          sml = 4.57 * tmx(j)
>           if (pmelt(j).le.10) then
>            a = 0.457
>         elseif (pmelt(j).ge.35) then
>              a = 3.66
>           else
>              a = 0.00549*pmelt(j)*pmelt(j) - 0.11887*pmelt(j) + 1.0973
>           endif
> c       sml = a * tmx(j)
>         sml = a * tx(j)
>           if (sml.gt.sno(j)) sml = sno(j)
>         else
>           sml = 0.
>         end if
>         if (tx(j).le.0) then
>                 snow = precip
>       else
>                 snow = 0
>         endif
69c88,106
<       end if
---
> c     add/sub aggregate snow fall and melt from effective precip
> c     and snow cover
>       precip = precip + sml -snow
>       sno(j) = sno(j) + snow - sml
>
> C Calculate percent melted for snowpack
>       if (sno(j).lt.0.0001.and.ida.gt.151) then
>         pmelt(j) = 0.
>         snomx(j) = 0.
>       else
>         if (sno(j).gt.snomx(j)) snomx(j) = sno(j)
>         if (snomx(j).le.0.) then
>           pmelt(j) = 0.
>       else
>           pmelt(j) = 100*(snomx(j) - sno(j))/snomx(j)
>       endif
```

< Original version

> New version

```
>       endif
> C Changes made in this subroutine 4/15/97 by Mike Hanratty to improve
the snowmelt
> C algorithm.
```

solp.f

No changes.

solt.f

No changes.

sort.f

No changes.

stat.f

```
Debugging
24,25c24,25
<       call sort(x,ii,xs)
<       call sort(y,ii,yys)
---
> c      call sort(x,ii,xs)
> c      call sort(y,ii,yys)
62,63c62,63
<         zzz = yys(i) - xs(i)
<         syb = syb + zzz * zzz
---
> c         zzz = yys(i) - xs(i)
> c         syb = syb + zzz * zzz
```

statcomp.f

No changes.

subbasin.f

```
Debugging
103c103,104
<         ssb(1) = ssb(1) + subp(j) * p2(j) * flu(j)
---
> c This would duplicate the summation done in clicon, line 216
> c         ssb(1) = ssb(1) + subp(j) * p2(j) * flu(j)
175d175
<         qdr = qdr - twl
```

sum.f

No changes.

surface.f

```
Debugging/snowmelt algorithm
71c71
<           call snom(j)
---
```

< Original version
> New version

```
>           if (sno(j).gt.0..or.precip.gt.0.) call snom(j)
```

swbl.f

No changes.

swu.f

No changes.

theta.f

No changes.

tran.f

No changes.

transfer.f

Changed subroutine name, transfer was a reserved word in my compiler

```
1c1
<       subroutine transfer(icode,ihout,inum1,inum2,inum3,rnum1,inum4)
---
>       subroutine transferw(icode,ihout,inum1,inum2,inum3,rnum1,inum4)
```

tstr.f

No changes.

ttcoef.f

No changes.

vald.f

Debugging

```
33c33,34
<       dimension ob(360), pd(360)
---
> c      dimension ob(360), pd(360)
>       dimension ob(j), pd(j)
```

varinit.f

No changes.

varout.f

No changes.

vbl.f

No changes.

virtual.f

Commented out some daily writes to save on disk space.

< Original version
> New version

```
264,266c264,266
<           write (77,5100) inum3, nbgis(j),inum3s(j), ida, precip, sml,
<      *        qd, wysb, eo, uw, yd / (100.*da*rnum1), yon, yph, yno3 +
<      *        ssfn, ysp
---
> c          write (77,5100) inum3, nbgis(j),inum3s(j), ida, precip,
sml,
> c     *        qd, wysb, eo, uw, yd / (100.*da*rnum1), yon, yph, yno3
+
> c     *        ssfn, ysp
271,279c271,279
<         write (3,5000) j, nbgis(j), nbigs(j), nmgt(j), ida, da *
<      *      flu(j), precip, snow, sml, qi, ssf, gwq(j), revap, gwseep,
<      *      gwchrg, wysb, sep, uw, qtl, yd / (100.*da*flu(j)), ev, sp,
<      *      rl, q1, pq, o, yp, ssub(22,inum1), ssub(23,inum1),
<      *      ssub(24,inum1), eo, ssub(26,inum1), ssub(27,inum1), (1.-
<      *      ws(j)), (1.-ts), (1.-strsn), (1.-strsp), yon, yph, yno3,
<      *      ssfn, ysp, uno3, percn, uap, rmn1tl, roctl,
<      *      fn(nro(j),nfert(j),j), fph(nro(j),nfert(j),j), wfx, wdntl,
<      *      hmntl, rwntl, hmptl, rmn2tl, rmptl, sw(j), dm(j), alai(j)
---
> c        write (3,5000) j, nbgis(j), nbigs(j), nmgt(j), ida, da *
> c     *      flu(j), precip, snow, sml, qi, ssf, gwq(j), revap,
gwseep,
> c     *      gwchrg, wysb, sep, uw, qtl, yd / (100.*da*flu(j)), ev,
sp,
> c     *      rl, q1, pq, o, yp, ssub(22,inum1), ssub(23,inum1),
> c     *      ssub(24,inum1), eo, ssub(26,inum1), ssub(27,inum1), (1.-
> c     *      ws(j)), (1.-ts), (1.-strsn), (1.-strsp), yon, yph, yno3,
> c     *      ssfn, ysp, uno3, percn, uap, rmn1tl, roctl,
> c     *      fn(nro(j),nfert(j),j), fph(nro(j),nfert(j),j), wfx,
wdntl,
> c     *      hmntl, rwntl, hmptl, rmn2tl, rmptl, sw(j), dm(j), alai(j)
```

volq.f

No changes.

washp.f

No changes.

water.f

No changes.

watqual.f

Point sources of beef manure (not used)

```
142a143,153
> c If there is runoff, assume point source from feedlots, etc.
> c This value assumes that a certain amount of mass is transported
> c by runoff, regardless of the amount of runoff
> c When "if" is commented out, we have direct deposit of beef manure
> c to the stream channel.
> c      if (j.eq.55) then
> c      if (ssb(3).gt.0.0) then
> c               pt_totn = 2.5
> c      else
>                 pt_totn = 0.0
```

< Original version
> New version

```
> c      endif
145,146c156,161
<          orgnin = 1000. * varoute(4,inum2) / varoute(2,inum2)
<          ammoin = 1000. * varoute(14,inum2) / varoute(2,inum2)
---
> c For each pair w/pt_totn, the first is the proportion for fresh beef
manure
> c and the second is the proportion for beef feedlot scrapings.
> c      orgnin = 1000.*(varoute(4,inum2)+0.65*pt_totn)/varoute(2,inum2)
>        orgnin = 1000. *(varoute(4,inum2)+0.9*pt_totn)/varoute(2,inum2)
> c
ammoin=1000.*(varoute(14,inum2)+0.3465*pt_totn)/varoute(2,inum2)
>
ammoin=1000.*(varoute(14,inum2)+0.05*pt_totn)/varoute(2,inum2)
148,150c163,168
<          nitratin = 1000. * varoute(6,inum2) / varoute(2,inum2)
<          orgpin = 1000. * varoute(5,inum2) / varoute(2,inum2)
<          dispin = 1000. * varoute(7,inum2) / varoute(2,inum2)
---
> c
nitratin=1000.*(varoute(6,inum2)+0.012*pt_totn)/varoute(2,inum2)
>
nitratin=1000.*(varoute(6,inum2)+0.17*pt_totn)/varoute(2,inum2)
> c         orgpin =
1000.*(varoute(5,inum2)+0.1*pt_totn)/varoute(2,inum2)
>          orgpin =
1000.*(varoute(5,inum2)+0.2*pt_totn)/varoute(2,inum2)
> c         dispin = 1000.
*(varoute(7,inum2)+0.2*pt_totn)/varoute(2,inum2)
>          dispin = 1000.
*(varoute(7,inum2)+0.3*pt_totn)/varoute(2,inum2)
152a171,180
> c         chlin = 1000. * varoute(13,inum2) / varoute(2,inum2)
> c         algin = 1000. * chlin / ai0
> c         orgnin = 1000. * (varoute(4,inum2)+pt_totn) /
varoute(2,inum2)
> c         ammoin = 1000. * varoute(14,inum2) / varoute(2,inum2)
> c         nitritin = 1000. * varoute(15,inum2) / varoute(2,inum2)
> c         nitratin = 1000. * varoute(6,inum2) / varoute(2,inum2)
> c         orgpin = 1000. * varoute(5,inum2) / varoute(2,inum2)
> c         dispin = 1000. * varoute(7,inum2) / varoute(2,inum2)
> c         cbodin = 1000. * varoute(16,inum2) / varoute(2,inum2)
> c         disoxin= 1000. * varoute(17,inum2) / varoute(2,inum2)
162a191
> c      wtemp(nreach) = 33.03 / (1+ exp(0.14 * (15.98-tx(nreach))))


Changed units for output from ppm to kg/day, added sediment yield.
388c417,424
<       if (ipdwql .ne. 0) then
---
>       if (ipdwql .ne. 0 .and. iopt.eq.nreach) then
> c       write (80,5000) nreach, nbgis(j), ida, wtemp(nreach),
> c     *chlin,chlout,orgnin,organicn(nreach),ammoin,
> c     *ammonian(nreach),nitritin,nitriten(nreach),nitratin,
> c     *nitraten(nreach),orgpin,organicp(nreach),dispin,
> c     *disolvp(nreach),cbodin,bod(nreach),soxy,disoxin,
> c     *disox(nreach),varoute(2,inum2),ttime
> c Output in kg/day rather than mg/L -- Mike H. 6/17/97
390,394c426,445
<      *chlin,chlout,orgnin,organicn(nreach),ammoin,
```

< Original version
> New version

```
<       *ammonian(nreach),nitritin,nitriten(nreach),nitratin,
<       *nitraten(nreach),orgpin,organicp(nreach),dispin,
<       *disolvp(nreach),cbodin,bod(nreach),soxy,disoxin,
<       *disox(nreach),varoute(2,inum2),ttime
---
>       *chlin * varoute(2,inum2) / 1000.,
>       *chlout * varoute(2,inum2) / 1000.,
>       *orgnin* varoute(2,inum2) / 1000.,
>       *organicn(nreach)* varoute(2,inum2) / 1000.,
>       *ammoin* varoute(2,inum2) / 1000.,
>       *ammonian(nreach)* varoute(2,inum2) / 1000.,
>       *nitritin* varoute(2,inum2) / 1000.,
>       *nitriten(nreach)* varoute(2,inum2) / 1000.,
>       *nitratin* varoute(2,inum2) / 1000.,
>       *nitraten(nreach) * varoute(2,inum2) / 1000.,
>       *orgpin * varoute(2,inum2) / 1000.,
>       *organicp(nreach) * varoute(2,inum2) / 1000.,
>       *dispin * varoute(2,inum2) / 1000.,
>       *disolvp(nreach) * varoute(2,inum2) / 1000.,
>       *cbodin * varoute(2,inum2) / 1000.,
>       *bod(nreach) * varoute(2,inum2) / 1000.,
>       *soxy * varoute(2,inum2) / 1000.,
>       *disoxin * varoute(2,inum2) / 1000.,
>       *disox(nreach) * varoute(2,inum2) / 1000.,
>       *varoute(2,ihout)/(da*1000),yd,ttime
398c449
<  5000 format ('REACH',2i4,i5,22E12.4)
---
>  5000 format ('REACH',2i4,i5,23E12.4)
```

watuse.f

No changes.

wetcon.f

```
Debugging
53a54,55
> c pq used for both wetland storage and wetland sediment inflow,  when
> c used as wetland storage, changed to twl - Mike H. 8/6/97
61c63
<             ssb(33) = ssb(33) + pq * flu(j)
---
>             ssb(33) = ssb(33) + twl * flu(j)
75c77,78
<             twl = pq
---
> c            twl = pq
>             qdr = qdr - twl
```

wetlan.f

```
Debugging
26c26
< c     yp          |               |pond sediment outflow
---
> c     yp          |               |wetland sediment outflow
30,35c30,34
< c     ev          |               |pond evaporation
```

< Original version
> New version

```
< c     sp          |               |pond seepage
< c     rl          |               |pond rainfall
< c     afp         |
< c     vv          |
< c     o           |
---
> c     ev          |(m^3)          |wetland evaporation
> c     sp          |(m^3)          |wetland seepage
> c     rl          |(m^3)          |wetland rainfall
> c     vv          |(m^3)          |wetland volume
> c     o           |(m^3)          |wetland outflow
37a37
> c     twl         |(mm)           |wetland storage (inflow - outflow)
52c52,53
<       ev = 6. * eo * sa
---
> c Use total pot. ET for evaporation - Mike H. 8/6/97
>       ev = 6. * pet(j) * sa
54,56c55,58
<       rl = precip * sa * 10.
<       afp = aff * fw(j)
<       q1 = qd * 10. * (afp/10.-sa)
---
> c Use only rainfall - Mike H. 8/6/97
>       rl = subp(j) * sa * 10.
> c Changed units calculation - Mike H. 8/6/97
>       q1 = qdr * 10. * (da*flu(j)*fp(j)-sa)
82c84,86
<         pq = (qd*afp-o) / aff
---
> c pq used for both wetland storage and wetland sediment inflow,  when
> c used as wetland storage, changed to twl - Mike H. 8/6/97
>         twl = (q1-o)/(da*flu(j)*10.)
```

writea.f

```
Debugging
134c134
<             if (mo.le.mo1) then
---
> c            if (mo.le.mo1) then Moved to simulate.f to reduce
spaghetti -- Mike H. 4/15/97
254c254,255
<             end if
---
> c            end if
> c              mo1 = mo  Moved to simulate.f to reduce spaghetti -
Mike H. 3/25/97
256d256
<           mo1 = mo
```

writeaa.f

```
Debugging
95c95
< c     vl          |m/day          |burial velocity
---
> c     vl          |m/day          |CN min   - Mike H. 3/26/97
```

< Original version

> New version

```
Added variables for output
378c378
<         do 290 k = 1, 7
---
>         do 290 k = 1, 12
381c381
<         write (kw,6400) j, (amo(j,k),k = 1,8)
---
>         write (kw,6400) j, (amo(j,k),k = 1,12), swmo(j), vrmo(j)
469c469
<  6400 format (i5,14f9.2)
---
>  6400 format (i5,14f9.4)
470a471,475
> c 6600 format (///,t17,'AVE MONTHLY BASIN
VALUES',/t19,'SNOW',t37,'SUB',
> c     *    t46,'WATER',/t4,'MO',t11,'R',t19,'FALL',t28,'SUR Q',t37,
> c     *    'SUR
Q',t46,'YIELD',t55,'ET',t66,'Y',t76,'EO',/t10,'(MM)',t19,
> c     *    '(MM)',t28,'(MM)',t37,'(MM)',t46,'(MM)',t55,'(MM)',t64,
> c     *    '(T/HA)',t75,'(MM)')
472,475c477,483
<       *    t46,'WATER',/t4,'MO',t11,'R',t19,'FALL',t28,'SUR Q',t37,
<       *    'SUR
Q',t46,'YIELD',t55,'ET',t66,'Y',t76,'EO',/t10,'(MM)',t19,
<       *    '(MM)',t28,'(MM)',t37,'(MM)',t46,'(MM)',t55,'(MM)',t64,
<       *    '(T/HA)',t75,'(MM)')
---
>       *
t46,'WATER',t117,'SOIL',t127,'RES'/t4,'MO',t11,'R',t19,'FALL',
>       *   t28,'SUR Q',t37,'SUR
Q',t46,'YIELD',t56,'ET',t65,'Y',t74,'EO',
>       *   t81,'ORG N',t91,'NO3',t99,'ORG P',t108,'SOL P',t117,'WATER',
>       *   t127,'VOL'/t10,'(MM)',t19,'(MM)',t28,'(MM)',t37,'(MM)',t46,
>       *   '(MM)',t55,'(MM)',t63,'(T/HA)',t73,'(MM)',t80,'(KG/HA)',t89,
>       *   '(KG/HA)',t98,'(KG/HA)',t107,'(KG/HA)',t117,'(MM)',t126,
>       *   '(HA-M)')


writed.f

Debugging
12c12
< c    ida          |              |day of simulation
---
> c    ida          |              |day of simulation (set to next day
before writed is called)
28c28
< c    ssb(104)     |(mm)          |reservoir volumes
---
> c    ssb(104)     |(mm)          |groundwater flow contribution to
streamflow

Reducing size of daily write
57,58c57,58
<           write (124,5000) iyr, ida, (concen(nbigs(j)),
<       *     flow(nbigs(j)), j=1,lubtot)
---
>           write (124,5000) iyr, ida-1, (concen(nbigs(j)),
> c     *     flow(nbigs(j)), j=1,lubtot)
```

< Original version
> New version

```
Debugging
68c68
<              write (kw,6200) ida, ssb(1), ssb(3), ssb(4), ssb(6),
---
>              write (kw,6200) ida-1, ssb(1), ssb(3), ssb(4), ssb(6),
91c91
<                if (iprp.ne.0) write (5,5200) ida, ' SOL', (ssb(k),k =
47,
---
>                if (iprp.ne.0) write (5,5200) ida-1, ' SOL',
(ssb(k),k=47,
93c93
<                if (iprp.ne.0) write (5,5300) ida, ' SOR', (ssb(k),k =
74,
---
>                if (iprp.ne.0) write (5,5300) ida-1, ' SOR',
(ssb(k),k=74,
106c106,107
< 6200   format(i5,13f7.2,2f5.2,1x,4f8.2)
---
> c6200   format(i5,13f7.2,2f5.2,1x,4f8.2)
> 6200   format (i5,6f7.3,f7.4,f7.2,5f7.4,2f5.3,1x,4f8.2)


writem.f

Debugging
106c106
<          if (mo.ne.mo1) then
---
> c         if (mo.ne.mo1) then          Moved to simulate.f to reduce
spaghetti - Mike H. 4/15/97
108,110c108,114
<              nd = nc(mo1+1)-nc(mo1)
<              if (iwst.ne.0) wpd(immo) = srch(2,iopt) * 86400.
<              if (isst.ne.0) spd(immo) = srch(2,iopt)
---
>              ind = nc(mo1+1)-nc(mo1)
> c Changed wpd to water yield rather than runoff in a reach -- Mike H.
4/15/97
> c Changed back 5/23/97
> c convert stream flow from m**3/s to mm/day.
>              if (iwst.ne.0) wpd(immo) = srch(2,iopt) * 86.400 / da
> c             if (iwst.ne.0) wpd(immo) = smm(6)
>              if (isst.ne.0) spd(immo) = srch(4,iopt)
114,116c118,120
<              smm(8) = smm(8) / float(nd)
<              smm(9) = smm(9) / float(nd)
<              smm(10) = smm(10) / float(nd)
---
>              smm(8) = smm(8) / float(ind)
>              smm(9) = smm(9) / float(ind)
>              smm(10) = smm(10) / float(ind)
121c125
<                if (srch(2,j).gt.0.0.and.srch(4,j).gt.50.) then
---
>                if (srch(2,j).gt.1.e-30.and.srch(4,j).gt.50.) then
127c131
<                if (sedcon.gt.200000.) sedcon = 200000.
---
>                if (sedcon.gt.200000.0) sedcon = 200000.0
```

< Original version
> New version

```
129,131c133,135
<                nd = nc(mo1+1)-nc(mo1)
<                srch(1,j) = srch(1,j) / nd
<                srch(2,j) = srch(2,j) / nd
---
>                ind = nc(mo1+1)-nc(mo1)
>                srch(1,j) = srch(1,j) / ind
>                srch(2,j) = srch(2,j) / ind
140,144c144,148
<                srch(10,j) = srch(10,j) / nd
<                srch(11,j) = srch(11,j) / nd
<                srch(12,j) = srch(12,j) / nd
<                srch(13,j) = srch(13,j) / nd
<                srch(14,j) = srch(14,j) / nd
---
>                srch(10,j) = srch(10,j) / ind
>                srch(11,j) = srch(11,j) / ind
>                srch(12,j) = srch(12,j) / ind
>                srch(13,j) = srch(13,j) / ind
>                srch(14,j) = srch(14,j) / ind
161c165,166
<      *             smm(104), smm(108), smm(108)
---
>      *             smm(104), smm(108)
> c    *             , smm(108)
224,226c229,231
<                  nd = nc(mo1+1)-nc(mo1)
<                  sres(1,j) = sres(1,j) / nd
<                  sres(2,j) = sres(2,j) / nd
---
>                  ind = nc(mo1+1)-nc(mo1)
>                  sres(1,j) = sres(1,j) / ind
>                  sres(2,j) = sres(2,j) / ind
237a243,247
>            amo(mo1,9) = amo(mo1,9) + smm(40)
>            amo(mo1,10) = amo(mo1,10) + smm(42) + smm(45)
>            amo(mo1,11) = amo(mo1,11) + smm(41)
>            amo(mo1,12) = amo(mo1,12) + smm(43)
>
280,281c290,291
<              call writea(j)
<         endif
---
> c          call writea(j) Moved to simulate.f to reduce spaghetti -
Mike H. 4/15/97
> c        endif
295c305,306
<  6200 format (i5,13f7.2,2f5.2,1x,4f8.2)
---
> c 6200 format (i5,13f7.2,2f5.2,1x,4f8.2)
>  6200 format (i5,6f7.2,f7.5,f7.2,4f7.5,f7.4,2f5.3,1x,4f8.2)
```

xmon.f

No changes.

ysed.f

No changes.

< Original version
> New version