PROJECT #: PO-094550

XWULQW'SELU WATERSHED SWAT-MODFLOW MODEL VALIDATION REPORT

Development and Optimization Analysis

Seonggyu Park, Ph.D.[1]
David Serrano Suarez, Ph.D.[2]
Tom Gleeson, Ph.D.[2]
Jaehak Jeong, Ph.D., P.E.[1]

[1]*Blackland Research & Extension Center, Texas A&M AgriLife Research, 702 E. Blackland Road Temple, TX 76502, USA*
[2]*University of Victoria, (add affiliation)*

Contents

Figures

Table

KOKSILAH MODEL DEVELOPMENT AND OPTIMIZATION ANALYSIS

Contact

Seonggyu Park, Ph.D.
Assistant Research Scientist
254.774.6032
seonggyu.park@brc.tamus.edu
https://blackland.tamu.edu/people/par
seonggyu/

1. OVERVIEW

1.1 Report Type

Project-specific, for project PO-094550, a Koksilaha SWAT-MODFLOW model development and optimization.

1.2 SWAT-MODFLOW version

SWAT-MODFLOW Sep 22, 2023 (1.1.3) version of SWAT-MODFLOW with an option to use enhanced irrigation management.

- The SWAT-MODFLOW model executable (swatmf_rel230922.exe), corresponding to its source code repository (https://github.com/spark-brc/SWAT-MODFLOW3/releases/tag/v1.1.3).

1.3 Optimization framework

We leveraged several open-source software tools to implement the uncertainty quantification (UQ), parameter estimation (PE), and sensitivity analysis (SA) workflow detailed in Appendix A. The swatmf Python package served as the central hub, controlling these tools and Python libraries, pyEMU and flopy. The source code repository for the optimization framework is available on GitHub: https://github.com/spark-brc/koksilah_smrt_model.

2. Model Construction

2.1 SWAT Model Construction

For the Koksilah SWAT model,

Software/Datasets

Table 1 Data sources used to construct the Koksilah SWAT model.

As part of the HRU development, the DEM was used to generate a slope map. SWAT allows multiple slope classes, and the slope map was divided into five classes using natural breaks, resulting in the following percent slope classes: <0.05, 0.05 – 1, 1 – 2, 2 – 5, and >5. HRU thresholds were set to zero for landuse, soils, and slope, resulting in 2,112 HRUs.

2.2 MODFLOW Model Construction

The Koksilah MODFLOW model has 4 layers with 68 rows and 94 columns, for a total of 25,568 grids. Of these grids, 14,784 are active, and 795 cells are used for river boundary where the SWAT river network overlays. The cells have uniform dimensions of 300 by 300 meters. The cell size was selected to reflect the density of input information (deep percolation and river stage) that the SWAT model passes to the MODFLOW model domain and monitoring and pumping well availabilities while providing adequate output resolution (spatial groundwater level and recharge variation). Due to the extremely thin top layers in the Koksilah MODFLOW model, adjusting riverbed bottom elevations in parameterization process is challenging. Consequently, a drain package (170 cells) was introduced to manipulate water removal rates that vary proportionally with the difference between the water table (river stage) and the drain elevation.

Pumping data

2.3 Development of the Koksilah SWAT-MODFLOW Model

Linking SWAT and MODFLOW can be performed through GIS application (Park and Bailey, 2017) or using graphical user interfaces, e.g., SWATMOD-Prep (Bailey et al., 2017) and QSWATMOD (Park et al., 2019). In this study, we used QSWATMOD (https://github.com/spark-brc/QSWATMOD2), a recent approach with the geographic information system (GIS) based plugin that includes sub-modules for exporting, plotting, and mapping the results. Through the linking process, 941 river-grid and 68,798 DHRUs-grid objects were generated, respectively.

3. Model Optimization

3.1 Parameterization

According to the characteristics of the Koksilah model domain and locations of observation, a zonal approach (zones of piecewise constancy) was used in spatial parameterization (Doherty et al., 2010), dividing the model domain into 12 different zones for the MODFLOW domain. After initial assessments of streamflow, groundwater levels, and their patterns, a total of 57 parameters were utilized for model optimization. These included 14 SWAT parameters, hydraulic conductivity, specific yield, and specific storage from 12 zones (total: 36) (Figure 1), as well as Riverbed conductance from 7 regionalized zones (Figure 2).

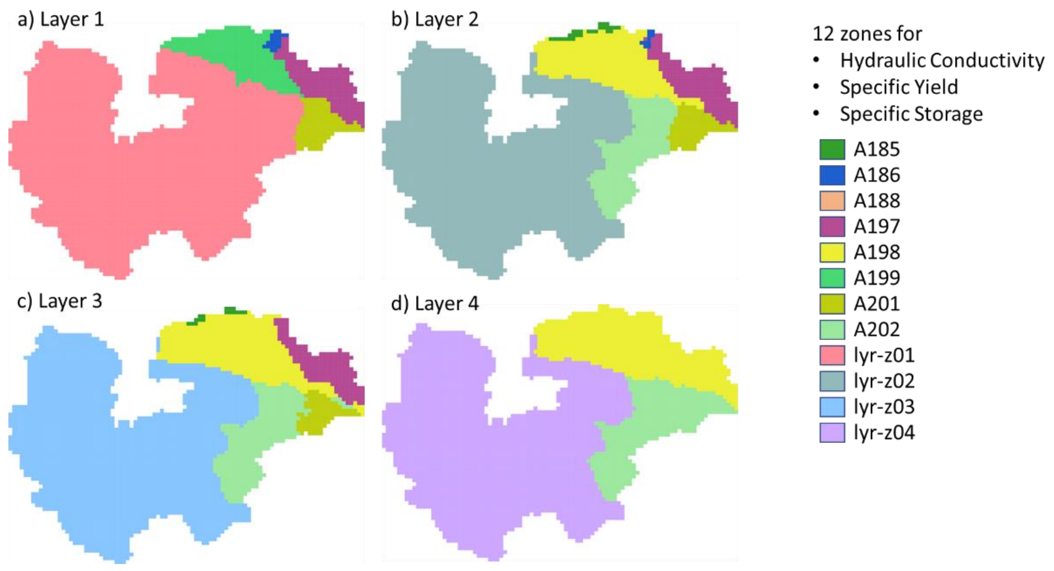

Figure 1. Zonal approach in spatial parameterization describing 12 different zones for geologic parameters (K, Sy, Ss)

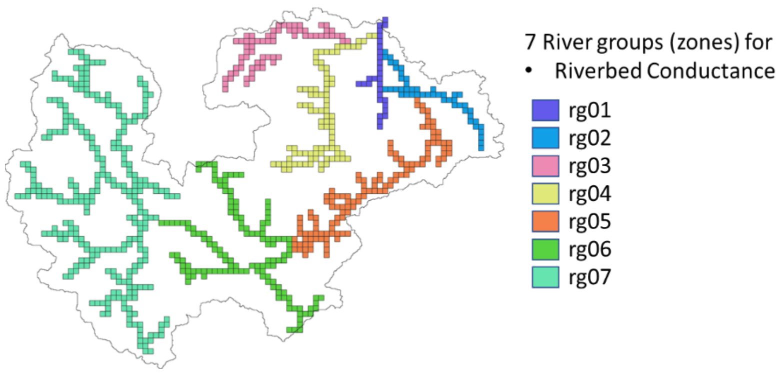

Figure 2. 7 regionalized stream networks for riverbed conductance

3.2 Data Availability

The Koksilah SWAT-MODFLOW model was calibrated using daily average stream discharge data from three gauging stations: RCH001 (2020-2022) and RCH003 (2013-2022). The significantly smaller dataset and lower discharge values at RCH001 (261 data points) compared to RCH003 (3,652 data points) pose challenges for model calibration, as accurately simulating low flow conditions is difficult and the limited data may contain noise. To improve low flow representation at RCH003, we assigned higher weights to streamflows below 0.2 m³/s by applying a reweighting factor of 1.5. The model was simultaneously calibrated using daily continuous groundwater levels from 3 locations (grid IDs: 249 in 2nd layer, 249 in 3rd layer and 1205 in 3rd layer) and static groundwater levels from 10 locations. Figure 3 illustrates the strategic placement of streamflow gauge stations and monitoring well locations within the study area.

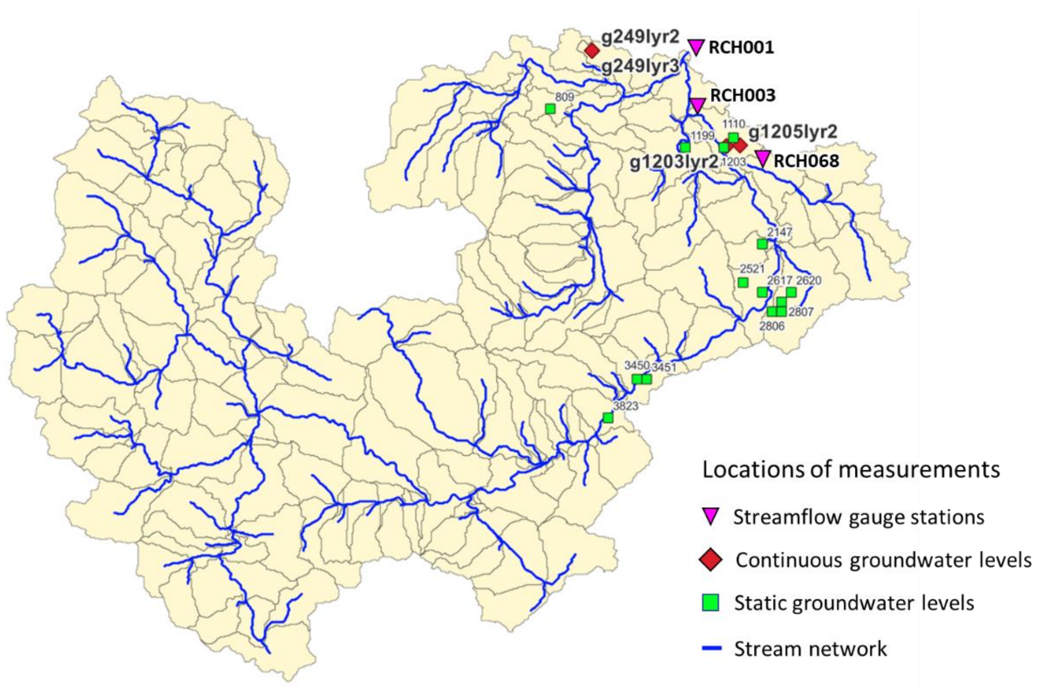

Figure 3. 7 Locations of streamflow and groundwater level monitoring data and stream network.

4. Results

4.1 Uncertainty analysis and calibration

For the Koksilah watershed's uncertainty analysis, we specified ten iterations and 3,000 model calls (300 realizations). The process converged successfully after nine iterations (2,700 model calls), meeting the termination criteria defined by White et al. (2020). To select the best-calibrated model from the posterior realizations, we employed a multi-variate objective function incorporating performance metrics such as R^2 and RMSE for depth to water, and NSE, R^2, and PBIAS for streamflow discharge (Table 2).

Figure 4 illustrates observed and best-estimated daily depth to water, along with prior and posterior prediction uncertainty bands for four locations. The gray lines represent the prior parameter ensembles (uncalibrated Monte Carlo results) with wider uncertainty, while the green band depicts the posterior ensemble, demonstrating a significant reduction in uncertainty. The calibration process appears to have been effective in reducing uncertainty and improving the model's performance at GRID249 in layer2 and layer3, and GRID1203 in layer3, but there are still challenges in accurately simulating groundwater levels at GRID1205.

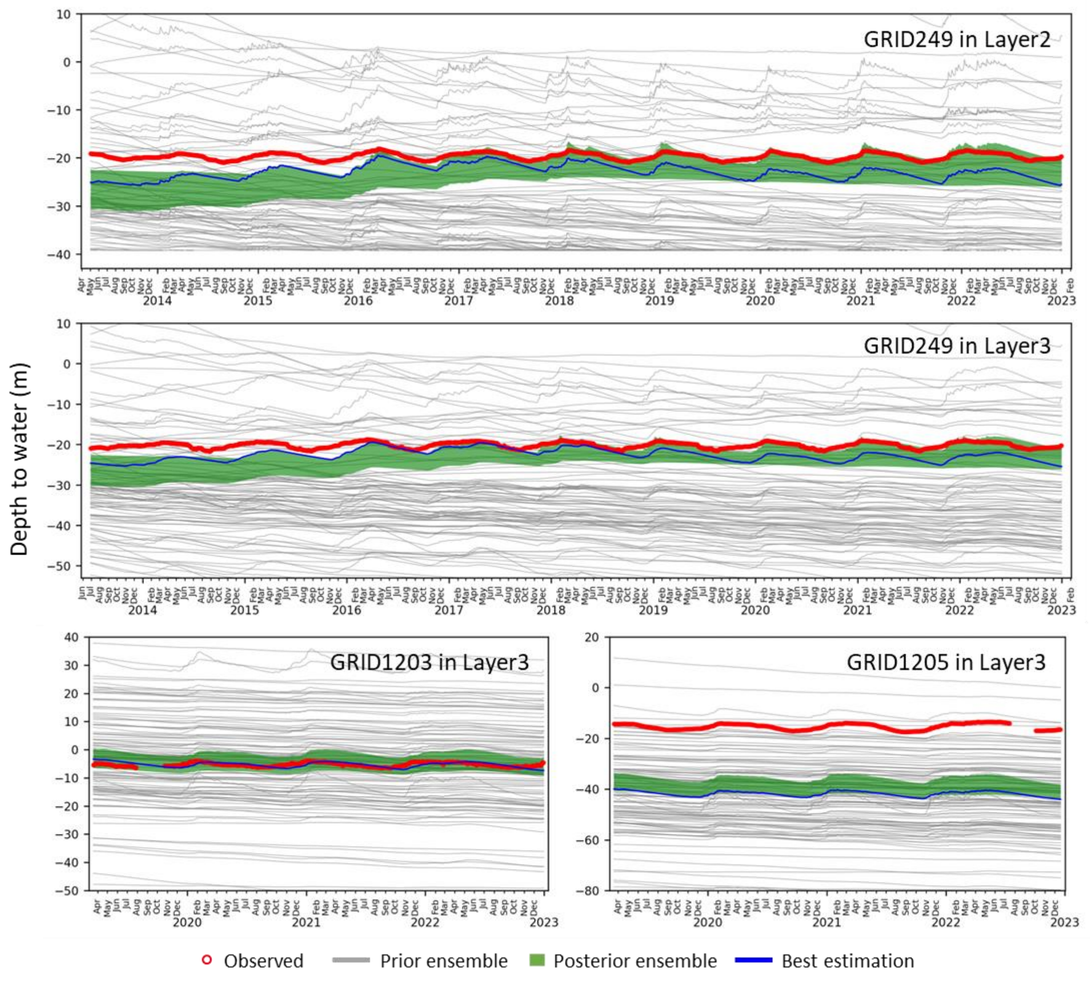

Figure 4. Observed (red) values vs simulated equivalent values for each of the prior (grey traces) and posterior (green traces) realizations with the best estimation.

Figure 5 presents a scatter plot comparing observed and simulated static depth to groundwater head values. Each data point is color-coded and labeled with the grid ID, layer number, and observation date. A green line represents the linear regression between the observed and simulated values, with the R² value (0.51) and RMSE (14.86) displayed in a green box.

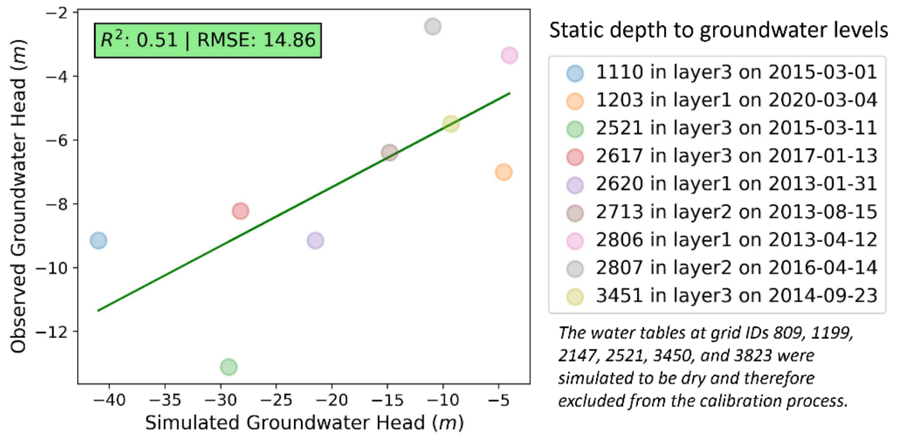

Figure 5. Comparison of Observed vs simulated static depth to groundwater levels from the best estimation

Figure 6 illustrates observed and best-estimated daily stream discharge, along with prior and posterior prediction uncertainty bands for three locations, showing the narrowing of prediction uncertainty bands from the prior (uncalibrated) to the posterior (calibrated) ensemble.

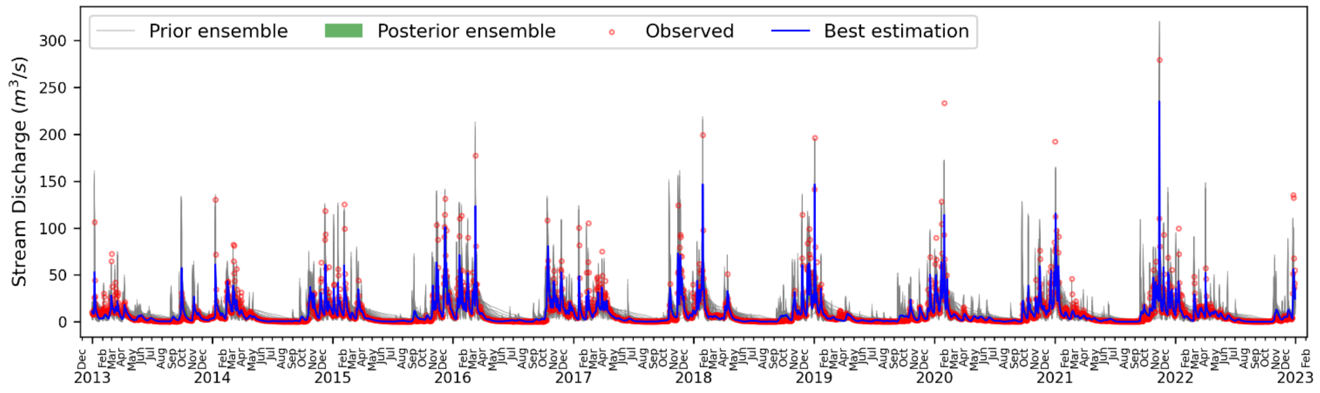

Figure 6. Observed (red) values vs simulated equivalent values in RCH003 with the prior (grey traces) and posterior (green band) realizations and the best estimation.

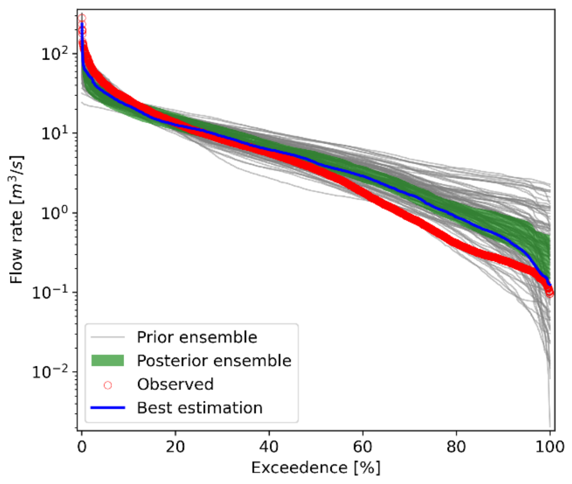

Figure 7. Comparison of Simulated and Observed Flow Durations with prior and posterior ensembles, and best estimation in RCH003

Table 2 Performance statistics: daily depth to waters at four locations, static depth to water levels and daily streamflow at three locations

	R2	RMSE	NSE	PBIAS
GRID249L2	0.37	3.39	-	-
GRID249L3	0.25	2.71	-	-
GRID1203L3	0.17	0.91	-	-
GRID1205L3	0.84	26.39	-	-
Static depth to water	0.51	14.86		
RCH001	0.44	3.35	-5.85	122.73
RCH003	0.62	11.50	0.61	-12.22

Table 3 Parameter values from Best estimation

Parameters	Types	Parameter ranges	Calibrated value
hk_a185	Hydraulic Conductivity	0.3 – 3700	36.3873
hk_a186		0.3 – 3700	31.9555
hk_a188		1 – 10000	878.56
hk_a197		0.01 – 200	1.71232
hk_a198		0.001 – 1	0.19782
hk_a199		0.001 – 1	0.4452
hk_a201		0.01 – 400	3.351358
hk_a202		0.001 – 100	0.206
hk_lyr01		0.001 – 1	0.01
hk_lyr02		0.0001 – 1	0.01
hk_lyr03		0.0001 – 1	0.05
hk_lyr04		0.00001 – 0.1	0.001
ss_a185	Specific Storage	1.00E-05 – 0.001	0.0001
ss_a186		1.00E-07 – 1.00E-05	6.50E-06
ss_a188		1.00E-07 – 1.00E-05	1.40E-06
ss_a197		1.00E-05 – 0.001	0.0001
ss_a198		1.00E-05 – 0.001	0.0001
ss_a199		1.00E-05 – 0.001	0.0001
ss_a201		1.00E-08 - 1.00E-06	1.00E-07
ss_a202		1.00E-06 – 1.00E-04	3.21E-05
ss_lyr01		1.00E-06 – 1.00E-04	1.00E-05
ss_lyr02		1.00E-06 – 1.00E-04	1.00E-05
ss_lyr03		1.00E-06 – 1.00E-04	1.00E-05
ss_lyr04		1.00E-06 – 1.00E-04	1.00E-05
sy_a185	Specific Yield	0.0001 – 0.6	0.01
sy_a186		0.0001 – 0.6	0.0254
sy_a188		0.0001 – 0.6	0.115
sy_a197		0.0001 – 0.6	0.01
sy_a198		0.0001 – 0.6	0.0138

sy_a199		0.0001 – 0.6	0.01
sy_a201		0.0001 – 0.6	0.013
sy_a202		0.0001 – 0.6	0.0142
sy_lyr01		0.0001 – 0.6	0.01
sy_lyr02		0.0001 – 0.6	0.01
sy_lyr03		0.0001 – 0.6	0.02
sy_lyr04		0.0001 – 0.6	0.01
rivcd_rg01	Riverbed Conductance	±90%	15.071
rivcd_rg02			18.366
rivcd_rg03			-41.655
rivcd_rg04			-52.4226
rivcd_rg05			-27.2768
rivcd_rg06			-0.5021
rivcd_rg07			-17.9047
alpha_bnk	rte	±90%	74.89
ch_k2	rte		89.46
ch_s2	rte		-0.1
chd	rte		84.5664
chw2	rte		69
cn2	mgt		0.001
epco	hru		-39.468
esco	hru		-51.26
hru_slp	hru		88.1
ov_n	hru		56.643
sol_awc	sol		74.211
sol_bd	sol		81.73
sol_k	sol		78.4
sol_z	sol		0.001

4.2 Sensitivity analysis using the Morris method

Figure 8 shows the results of a Morris sensitivity analysis, which is a method for assessing the relative importance of different parameters in a model. The x-axis represents the mean elementary effect (μ*), which measures the average impact of a parameter on the model output. The y-axis represents the sigma (σ) value, which measures the degree of nonlinearity or factor interaction associated with the parameter.

The results show that the parameters with the highest μ* values greater than 10 are curve number (cn2, swat), specific yield (sy_a198, zone: a198) and hydraulic conductivity (hk_a199, zone: a199), indicating that these parameters have the greatest impact on the model outputs, stream discharge and groundwater levels. The σ values for these parameters are also relatively high, suggesting that they have a significant degree of nonlinearity or factor interaction.

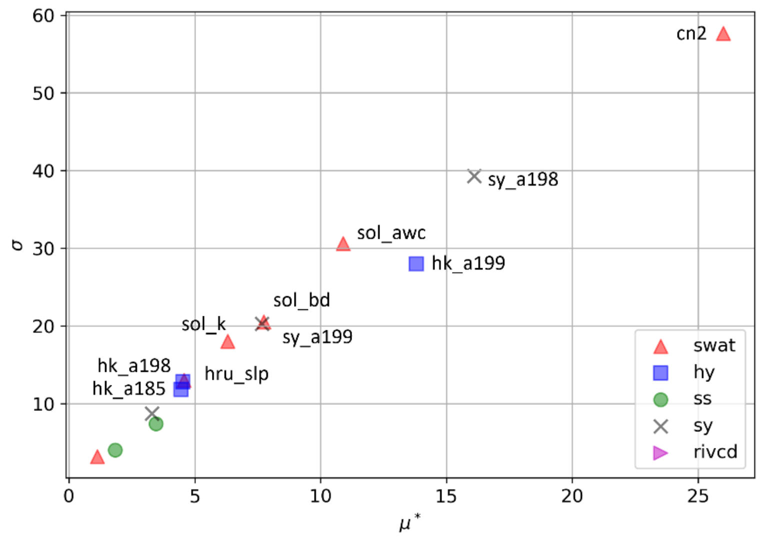

Figure 8. Parameter Sensitivity Analysis: Morris Screening Result: Only the most sensitive parameters are labeled for clarity. The σ values indicate the degree of nonlinearity or factor interaction, while μ represents the sensitivity measure.*

4.3 Spatiotemporal hydrological response

Groundwater Table Depths

Figure 9 depicts the average monthly groundwater recharge rates for March, June, September, and December between 2013 and 2022. The depth to water levels varies significantly throughout the year and across the study area. In March, water levels are generally higher in the northern and eastern regions. By June, overall water levels decline, though some areas in the south show elevated levels. In September, water levels rise again in the northern and eastern regions, while in December, low water levels are observed throughout the entire study area. Contours are included in the maps to show the spatial variation in depth to water levels. The contours are labeled with different depths, ranging from -300 m to 0 m.

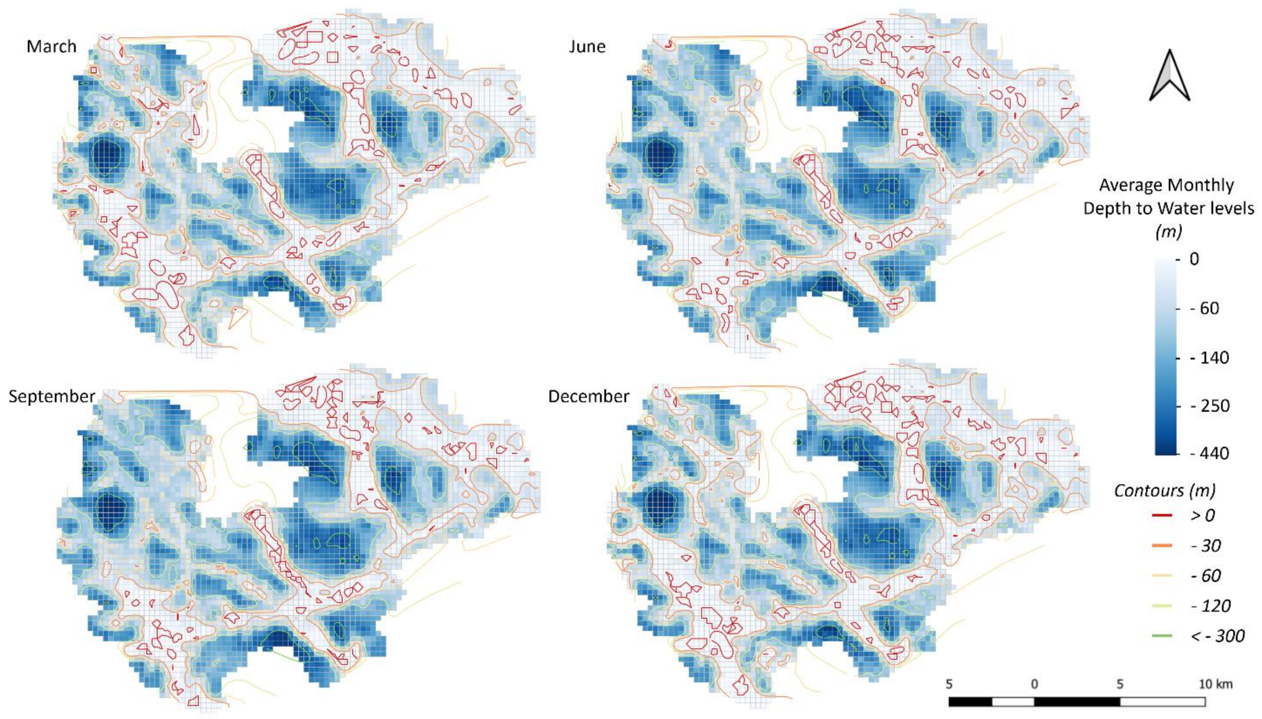

Figure 9. Seasonal Depth to Water Levels in the Koksilah Watershed

Groundwater Recharge

Figure 10 shows that average monthly groundwater recharges in March, June, September, and December from 2013 ~ 2022. The results show groundwater recharge varies significantly throughout the year and across the study area. In March, recharge rates are generally higher in the northern and eastern parts of the study area. In June and September, recharge rates are lower overall, but there are some areas with higher recharge rates in the northern and eastern parts of the study area. In December, recharge rates are again higher in the northern and eastern parts of the study area.

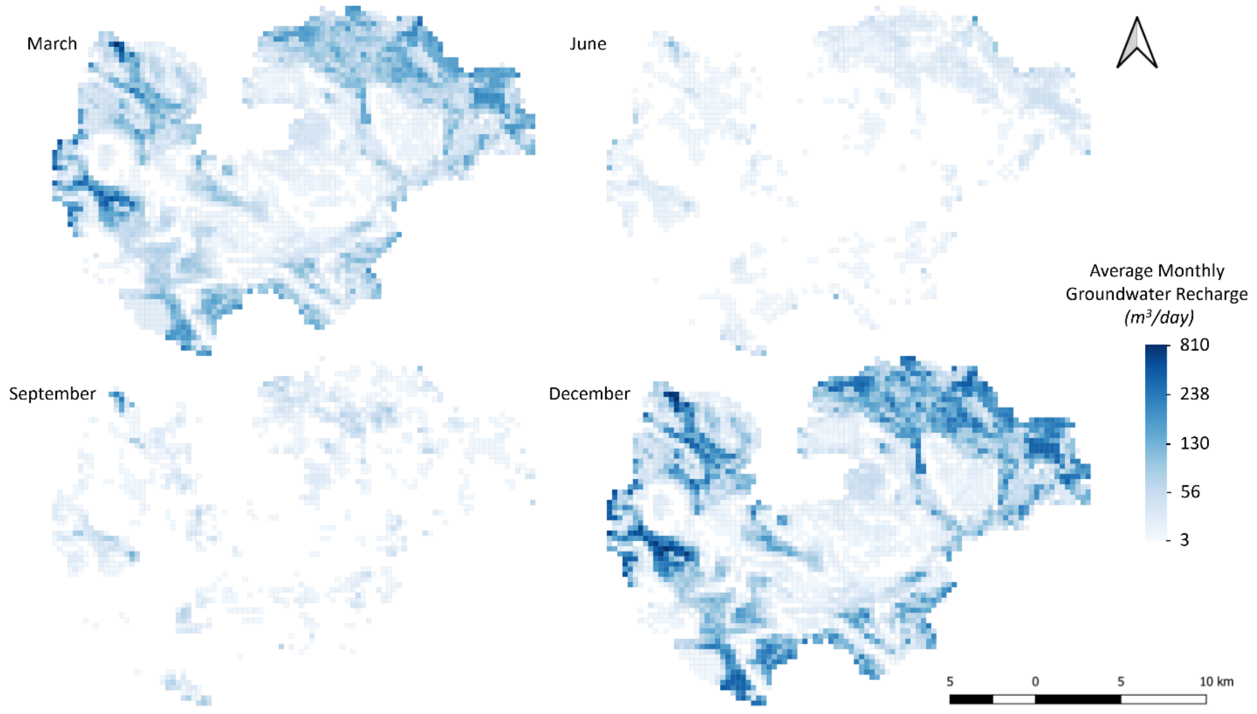

Figure 10. Spatial and temporal variability of groundwater recharge in the Koksilah Watershed

Groundwater-Surface Water Interaction

Figure 11 illustrates the seasonal variation in groundwater-surface water interactions from 2013 to 2022, focusing on March, June, September, and December. In March and December, groundwater discharge to streams predominated in most areas, particularly in the northern and eastern regions. In June and September, groundwater is still flowing to the surface water in some areas, but there are also areas where surface water is flowing to the groundwater.

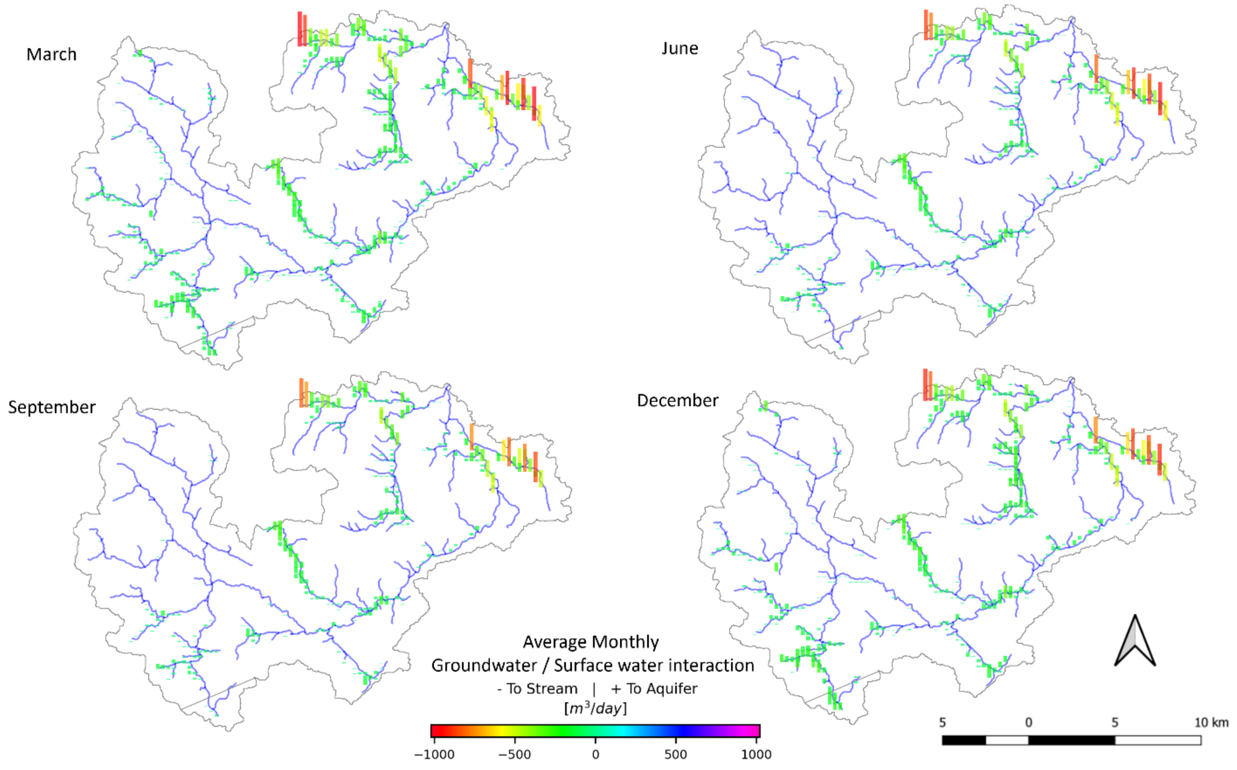

Figure 11. Spatial and temporal variability of groundwater-surface water interaction in the Koksilah Watershed

Water Balance

Figure 12 shows that the precipitation fluctuates throughout the year, with higher levels in the spring and winter months and lower levels in the summer and fall months. The soil water content generally increases after periods of high precipitation and decreases during dry periods. Surface runoff and lateral flow are also higher during periods of high precipitation. Groundwater flows to the stream and seepage from the stream to the aquifer both vary depending on the water levels in the stream and the aquifer. Deep percolation to the aquifer is generally low, indicating that most of the water that infiltrates the soil is retained in the upper layers. The groundwater volume shows a general trend of increase over the time period analyzed, with some fluctuations due to variations in precipitation and other water balance components.

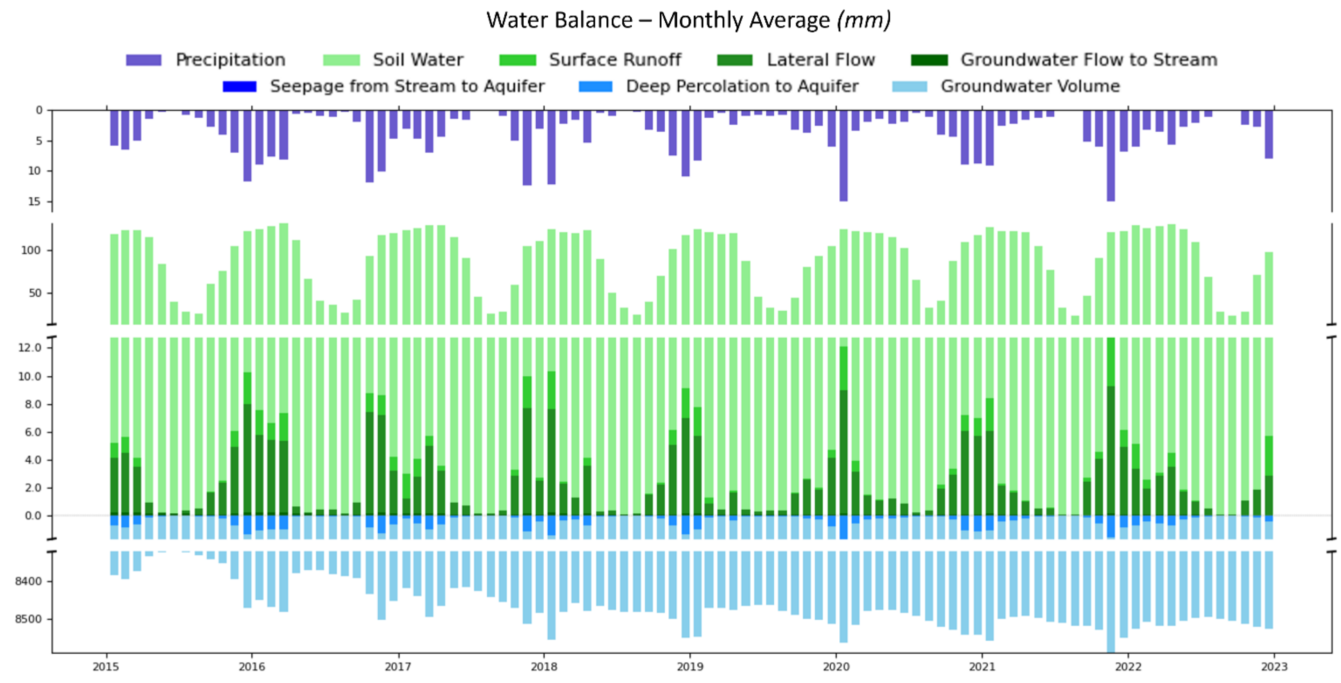

Figure 12. Monthly Average Water Balance in the Koksilah Watershed

References

Arnold, J. G., & Fohrer, N. (2005). SWAT2000: Current capabilities and research opportunities in applied watershed modelling. *Hydrological Processes*, *19*(3), 563–572. https://doi.org/10.1002/hyp.5611

Bailey, R., Rathjens, H., Bieger, K., Chaubey, I., & Arnold, J. (2017). SWATMOD-Prep: Graphical User Interface for Preparing Coupled SWAT-MODFLOW Simulations. *JAWRA Journal of the American Water Resources Association*, *53*(2), 400–410. https://doi.org/10.1111/1752-1688.12502

Bailey, R. T., Wible, T. C., Arabi, M., Records, R. M., & Ditty, J. (2016). Assessing regional-scale spatio-temporal patterns of groundwater-surface water interactions using a coupled SWAT-MODFLOW model: SWAT-MODFLOW Coupled Watershed Modelling. *Hydrological Processes*. https://doi.org/10.1002/hyp.10933

Blasone, R.-S., Vrugt, J. A., Madsen, H., Rosbjerg, D., Robinson, B. A., & Zyvoloski, G. A. (2008). Generalized likelihood uncertainty estimation (GLUE) using adaptive Markov Chain Monte Carlo sampling. *Advances in Water Resources*, *31*(4), 630–648. https://doi.org/10.1016/j.advwatres.2007.12.003

Doherty, J. E., Hunt, R. J., & Tonkin, M. J. (2010). *Approaches to Highly Parameterized Inversion* (Scientific Investigations Report) [Scientific Investigations Report].

Hunt, R. J., Luchette, J., Schreuder, W. A., Rumbaugh, J. O., Doherty, J., Tonkin, M. J., & Rumbaugh, D. B. (2010). Using a Cloud to Replenish Parched Groundwater Modeling Efforts. *Ground Water*, *48*(3), 360–365. https://doi.org/10.1111/j.1745-6584.2010.00699.x

Liu, W., Park, S., Bailey, R. T., Molina-Navarro, E., Andersen, H. E., Thodsen, H., Nielsen, A., Jeppesen, E., Jensen, J. S., Jensen, J. B., & Trolle, D. (2019). *Comparing SWAT with SWAT-MODFLOW hydrological simulations when assessing the impacts of groundwater abstractions for irrigation and drinking water* [Preprint]. Catchment hydrology/Modelling approaches. https://doi.org/10.5194/hess-2019-232

Neitsch, S. L., Arnold, J. G., Kiniry, J. R., & Williams, J. R. (2005). *SOIL AND WATER ASSESSMENT TOOL THEORETICAL DOCUMENTATION*. 494.

Niswonger, R. G., Panday, S., & Ibaraki, M. (2011). MODFLOW-NWT, a Newton formulation for MODFLOW-2005. *US Geological Survey Techniques and Methods*, *6*(A37), 44.

Park, S. (2018). *ENHANCEMENT OF COUPLED SURFACE / SUBSURFACE FLOW MODELS IN WATERSHEDS: ANALYSIS, MODEL DEVELOPMENT, OPTIMIZATION, AND USER ACCESSIBILITY*.

Park, S., & Bailey, R. T. (2017). *SWAT-MODFLOW Tutorial—Documentation for Preparing Model Simulations*.

Park, S., Nielsen, A., Bailey, R. T., Trolle, D., & Bieger, K. (2019). A QGIS-based graphical user interface for application and evaluation of SWAT-MODFLOW models. *Environmental Modelling & Software*, *111*, 493–497. https://doi.org/10.1016/j.envsoft.2018.10.017

Perez, F., & Granger, B. E. (2007). IPython: A System for Interactive Scientific Computing. *Computing in Science & Engineering*, *9*(3), 21–29. Computing in Science & Engineering. https://doi.org/10.1109/MCSE.2007.53

Van Rossum, G., & Drake, F. L. (2009). *Python 3 reference manual*. CreateSpace.

Welter, D. E., White, J. T., Hunt, R. J., & Doherty, J. E. (2015). Approaches in highly parameterized inversion—PEST++ Version 3, a Parameter ESTimation and uncertainty analysis software suite optimized for large environmental models. In *Approaches in highly parameterized inversion—PEST++ Version 3, a Parameter ESTimation and uncertainty analysis software suite optimized for large environmental models* (USGS Numbered Series 7-C12; Techniques and Methods, Vols. 7-C12, p. 64). U.S. Geological Survey. https://doi.org/10.3133/tm7C12

White, J. T. (2018). A model-independent iterative ensemble smoother for efficient history-matching and uncertainty quantification in very high dimensions. *Environmental Modelling & Software*, *109*, 191–201. https://doi.org/10.1016/j.envsoft.2018.06.009

White, J. T., Fienen, M. N., & Doherty, J. E. (2016). A python framework for environmental model uncertainty analysis. *Environmental Modelling & Software*, *85*, 217–228. https://doi.org/10.1016/j.envsoft.2016.08.017

White, J. T., Hunt, R. J., Fienen, M. N., & Doherty, J. E. (2020). Approaches to highly parameterized inversion: PEST++ Version 5, a software suite for parameter estimation, uncertainty analysis, management optimization and sensitivity analysis. In *Techniques and Methods* (7-C26). U.S. Geological Survey. https://doi.org/10.3133/tm7C26

Wu, B., Zheng, Y., Tian, Y., Wu, X., Yao, Y., Han, F., Liu, J., & Zheng, C. (2014). Systematic assessment of the uncertainty in integrated surface water-groundwater modeling based on the probabilistic collocation method. *Water Resources Research*, *50*(7), 5848–5865. https://doi.org/10.1002/2014WR015366

Appendix A: Model Construction

SWAT

SWAT is a physically-based, semi-distributed hydrologic model that can operate at annual, monthly, and daily time steps. SWAT was originally designed to use in large ungauged agricultural basins to assess the relative impact of land management decisions on a variety of water quality and quantity parameters (Arnold & Fohrer, 2005; Neitsch et al., 2005). While originally intended for use in the United States, the model has been adapted for use worldwide including throughout Africa. Strengths of the SWAT model are that it can be set up using readily available global data and then be used to rapidly assess and develop an understanding of long-term impacts on annual or seasonal water balance resulting from potential climate or management changes.

SWAT implements the well-established Soil Conservation Service Curve Number (SCS – CN) method to calculate runoff using an empirical water balance relationship:

$$Q = \frac{(P - 0.2S)^2}{P + 0.8S} \tag{1}$$

where, Q is the direct runoff (mm); P is the total rainfall (mm); and S = 1000/CN, with CN (curve number) related to soil and land cover conditions, and commonly estimated from published tables.[1]

Erosion and sediment yield in SWAT …

Due to its deterministic nature, each successive SWAT run that uses a given set of inputs will produce the exact same outputs each time the model is run. If a user modifies an input, such as climate or land management, however, the output may change. This allows users to isolate the response to that specific given change. In this way, SWAT is an excellent model for exploring alternative land management scenarios or interventions and how they are likely to modify the water balance, and this is the most basic function of the SWAT model.

MODFOW

MODFLOW is a modular finite-difference groundwater flow model developed and maintained by the U.S. Geological Survey (USGS). The model solves the following conservation of mass equations, wherein groundwater mass is accounted for in representative volumes of an unconfined aquifer:

$$\frac{\partial}{\partial x}(\mathrm{F}_s K_{xx}\frac{\partial h}{\partial x}) + \frac{\partial}{\partial y}(\mathrm{F}_s K_{yy}\frac{\partial h}{\partial y}) + \frac{\partial}{\partial z}(\mathrm{f}(\mathrm{F}) K_{zz}\frac{\partial h}{\partial z}) + W = \varphi\frac{\partial \mathrm{F}_s}{\partial t} + \mathrm{F}_s S_s\frac{\partial h}{\partial t} \tag{2}$$

where x, y, z are the three dimensions in the aquifer; h is groundwater head (L); K is hydraulic conductivity (L/T), Ss is specific storage (1/T), φ is porosity (L^3/L^3) and is assumed equal to specific yield Sy (L^3/L^3); F_S is the fraction of the cell thickness that is saturated; and f(F) is a function of F_S set to 1 for (Niswonger et al., 2011). Equation 3 represents a water balance for a given volume of the aquifer, with groundwater flow in/out in the x, y, and z directions, sources

[1] Full theoretical documentation on SWAT can be found at https://swat.tamu.edu/media/99192/swat2009-theory.pdf and https://swat.tamu.edu/docs/.

and sinks (e.g., recharge, pumping, discharge to streams, seepage from streams) represented by W, and storage changes represented by the terms on the right-hand side.

The aquifer domain is divided into cubic grid cells, and the water balance of Equation 3 is used to update groundwater storage and associated groundwater head *h* in each cell during each time step of a transient simulation. The updated *h* values for each cell are then used to estimate groundwater flow direction and groundwater travel time throughout the aquifer. The minimum input data required to set up MODFLOW grids consist of geological formations (land surface elevation, depth to bedrock, boundary condition) and corresponding hydraulic conductivity (K) and aquifer storage properties (S_s and S_y).

SWAT-MODFLOW

The coupled SWAT-MODFLOW model from the source codes of SWAT (Revision 591) and MODFLOW-NWT was combined as a single FORTRAN executable file, developed by Bailey et al. (2016). Within this coupled modeling framework, MODFLOW-NWT code is called as a sub-module of SWAT code. SWAT simulates land surface processes, crop growth, in-stream processes, and soil zone processes. In contrast, MODFLOW simulates three-dimensional groundwater flow and all associated sources and sinks (e.g., recharge, pumping, discharge to tile drains, and interaction with stream network). The daily based-simulation process of the linked SWAT-MODFLOW model is to pass HRU-calculated deep percolation as recharge to the grid cells of MODFLOW, then pass MODFLOW-calculated groundwater–surface water interaction fluxes to the stream channels of SWAT. With this approach, SWAT calculates the volume of overland flow and soil lateral flow to streams, MODFLOW calculates the volume of groundwater discharge to streams, and then SWAT routes the water through the watershed's stream network. Flux interaction between groundwater and surface water is simulated using the River package of MODFLOW and simulated stream stages. Darcy's law is used to calculate the volumetric flow interactions of water through the cross-sectional flow area between the aquifer and the stream channel using the River package of MODFLOW and simulated stream stages of SWAT.

$$Q_{leak} = K_{bed} \times (L_{str} \times P_{str}) \times \left(\frac{h_{str} - h_{aq}}{z_{bed}}\right) \tag{3}$$

where K_{bed} is riverbed hydraulic conductivity [L/T], L_{str} is the length of the stream [L], P_{str} is the wetted perimeter of the stream [L], h_{str} is river stage [L], h_{gw} is the hydraulic head of groundwater [L], and z_{bed} is the thickness of the riverbed [L].

Appendix B: Model Optimization

Description of Optimization Framework

Several open-source software tools were used to implement the uncertainty quantification (UQ), parameter estimation (PE), and sensitivity analysis (SA) workflow.

- The Python package swatmf (Park et al., 2024 in review, https://doi.org/10.5281/zenodo.11194863) was used to programmatically load, process, and manipulate an existing SWAT model and utilize PEST utilities;
- The python package pyEMU (White et al., 2016, https://doi.org/10.1016/j.envsoft.2021.105022) was used to programmatically construct a high-dimensional PEST interface (Doherty, 2015b) around the forward model and the generate the prior parameter ensembles;
- The iterative ensemble smoother PESTPP-IES (White, 2018, https://doi.org/10.3133/tm7C26) was used to evaluate the prior parameter ensembles (for UQ) and to also perform formal, high-dimensional PE;
- The PESTPP-SEN tool within the PEST++ environment is utilized to generate parameter values, update model files for the Koksilah SWAT-MODFLOW model, run the model simulations and compute sensitivity indices for the Morris method;
- A web-based interactive development environment called Jupyter notebooks (Perez & Granger, 2007) was utilized along with the above-mentioned python packages and software tools to construct an interactive platform that allows for analyzing and documenting exploratory parameter and uncertainty analysis workflows (https://github.com/spark-brc/swatmf_wf).

Integrated hydrologic models commonly include substantial uncertainties with respect to input data, forcing data, initial and boundary conditions, model structure, and parameter non-uniqueness due to a lack of data and poor knowledge of hydrological response mechanisms (Blasone et al., 2008; Wu et al., 2014). Besides, the application of parameter estimation (PE) and uncertainty analysis (UA) techniques to integrated surface water-groundwater models is difficult because of

- larger number of potentially correlated and often insensitive parameters
- longer model runtimes
- large, transient observation datasets with correlated and often redundant information
- more complexity of preparation, implementation, and post-processing of the analysis (no systematic approach)

Building upon previous work (Liu et al., 2019; Park, 2018), a single, automated calibration method was developed to estimate both land surface (SWAT) and subsurface (MODFLOW) parameters simultaneously. This method leverages PEST (Doherty et al., 2010) for optimization, Python scripting (Van Rossum & Drake, 2009) for automation, and batch files for streamlined execution. The recently developed swatmf Python package (Park et al., 2024, in review; https://doi.org/10.5281/zenodo.11194863) simplifies the workflow by programmatically handling model data, constructing PEST++ interface, utilizing PEST++ utilities, and facilitating visualization of optimization results (Figure 4).

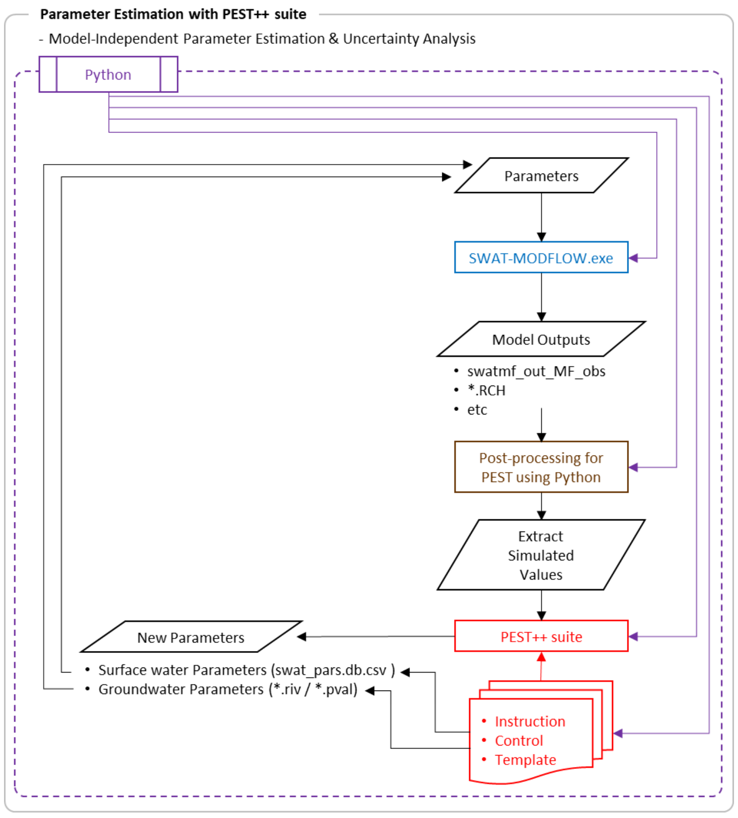
Parameter Estimation with PEST++ suite
- Model-Independent Parameter Estimation & Uncertainty Analysis
Python
Parameters
SWAT-MODFLOW.exe
Model Outputs
• swatmf_out_MF_obs
• *.RCH
• etc
Post-processing for PEST using Python
Extract Simulated Values
PEST++ suite
New Parameters
• Surface water Parameters (swat_pars.db.csv)
• Groundwater Parameters (*.riv / *.pval)
• Instruction
• Control
• Template

Figure B1. Schematic diagram for reproducible parameter estimation framework for both land surface and hydrogeological parameters.

The PEST++ suite TCP/IP-based parallel run manager is based on the PANTHER run manager (Welter et al., 2015), using the concepts of Hunt et al. (2010), which tolerates model run failures and includes multithreaded agents so that unneeded model runs can be preempted and runs allocated to lost agents can be efficiently rescheduled elsewhere (White et al., 2020). A web-based interactive development environment called Jupyter notebooks (Perez & Granger, 2007) was utilized along with the above-mentioned python packages and PEST++ suite to construct an interactive platform that allows for analyzing and documenting exploratory parameter and uncertainty analysis workflows. This technique can help improve the transparency and reproducibility of uncertainty analysis. The complete workflows for parameter estimation and uncertainty analyses of the Koksilah SWAT-MODFLOW model are documented in the Jupyter notebooks that are available in the Koksilah SWAT-MODFLOW repository (https://github.com/spark-brc/koksilah_smrt_model.git).

Implementation and workflow

Note that the online code repository (https://github.com/spark-brc/koksilah_smrt_model.git) for koki-swatmf includes Jupyter notebooks that provide more detailed descriptions for the arguments, as well as provide working examples of the analyses discussed herein. To run the optimization framework locally on your desktop, you'll need to install Miniconda and set up an environment first. Specific instructions for the workflow are provided within the repository (refer to steps marked with ">" for command prompt and ">>>" for Python code).

1. If you don't already have conda installed, please download Miniconda for your operating system from https://conda.io/en/latest/miniconda.html (choose the latest version for your operating system, 64-bit). You should not need elevated rights to install this.
2. Run the installer and select "only my user" when prompted. This will allow you to work with your python installation directly.
3. After installation, go to the START menu and select "Miniconda Prompt" to open a DOS box.
4. Using the cd command in the Miniconda DOS box, navigate to the location where you have environment.yml the file and type:

```
> conda env create -f environment.yml
```

5. After your virtual environment setup is complete, change the environment to koki-swatmf:

```
> conda activate koki-swatmf
```

6. Launch jupyter notebook

```
> jupyter notebook
```

The following block of code demonstrates the process of establishing zonal parameter estimation for the Koksilah SWAT-MODFLOW model.

1. import the `swatmf_pst_utils` from `swatmf` module into the current workflow

   ```
   >>> from swatmf import swatmf_pst_utils
   ```

2. initial setup for parameter estimation and create the "swatmf.con" file

   ```
   >>> swatmf_pst_utils.create_swatmf_con(
       prj_dir,
       swatmf_model,
       swat_model,
       sim_start,
       warmup,
       cal_start,
       cal_end,
       subs=None,
       riv_parm=None,
       time_step=None,
       pp_included=None,
       grids_lyrs=None,
       avg_grids=None,
       static_dtw=False,
             )
   ```

   - prj_dir : project folder for optimization. Note that all original files and folders are copied to this directory according to the optimization directory structure (string).
   - swatmf_model: SWAT-MODFLOW model folder that includes SWAT and MODFLOW models, and all necessary linkage files (string).
   - swat_model: only SWAT model (string).
   - sim_start: simulation start date (string), '1/1/2010'
   - warmup: warm up period (integer), 3
   - cal_start: calibration start date (string), '1/1/2013'
   - cal_end: calibration end date (string), '12/31/2022'
   - subs: reach indices for simulated stream discharge values that will be calibrated to match targeted observed stream discharge data (list), [1, 3, 68]
   - grids_lyrs: grid and layer indices to extract simulated groundwater levels that will be calibrated to match targeted observed groundwater level data (list), ["249lyr2", "249lyr3", "1205lyr3", "1203lyr3"]
   - time_step: model simulation step (string), 'day'
   - riv_parm: river group indices to adjust river parameters (riverbed conductance and riverbed bottom elevation, list), ["rg01", "rg02", "rg03", "rg04", "rg05", "rg06", "rg07"]
   - static_dtw: grid and layer indices to extract simulated depth to water levels that will be calibrated to match targeted observed static depth to water values (boolean), static_dtw=True, it will read the "dtw_static.obd.csv" file that contains 'grid_id', 'layer', 'static_dtw' (observed depth to water), and 'date'.

3. copy all necessary files to your working directory

   ```
   >>> swatmf_pst_utils.init_setup(prj_dir, swatmf_model, swat_model)
   ```

4. Build template files

```
>>> import pyemu
>>> pyemu.utils.gw_utils.modflow_pval_to_template_file(pval_file)
```

- `pyemu.utils.gw_utils.modflow_pval_to_template_file(pval_file)` read pval_file from MODFLOW model and create *.tpl file
- `pval_file`: MODFLOW parameter file (string), “koki.pval_adj”

```
>>> from swatmf.utils.swat_configs import SwatEdit
>>> m1 = SwatEdit(os.getcwd())
>>> m1.create_swat_pars_cal()
```

- `from swatmf.utils.swat_configs import SwatEdit` imports SwatEdit from the swatmf module into the current workflow.
- `m1 = SwatEdit(os.getcwd())` initiates m1 (model) variable with the current working directory.
- `m1.create_swat_pars_cal()` reads the SWAT model parameter database, the “swat_pars.db.csv” file and select parameters marked by flag , creates the "swat_pars.cal" model parameter input file and "swat_pars.cal.tpl" template file.

```
>>> from swatmf.utils.mf_configs import mfEdit
>>> mf = mfEdit(os.getcwd())
>>> mf.create_riv_par(
             riv_parm, chg_type=None, rivcd=None, rivbot=None, val=None
              )
```

- `from swatmf.utils.mf_configs import mfEdit` imports mfEdit from the swatmf module into the current workflow.
- `mf = mfEdit(os.getcwd())` initiates mf (model) variable with the current working directory.
- `m1.create_swat_pars_cal()` reads the existing the MODFLOW river package and creates the "mf_riv.par" model parameter input file and " mf_riv.par.tpl" template file.

5. Build instruction files

Note that you must run the model before building instruction files to get full simulation outputs. Refer to previously defined arguments.

```
>>> swatmf_pst_utils.extract_day_stf(
       subs, sim_start, warmup, cal_start, cal_end
       )
>>> swatmf_pst_utils.stf_obd_to_ins(
       stf_extracted_file, colname,
       cal_start, cal_end
       )
```

- `swatmf_pst_utils.extract_day_stf()` extracts a daily simulated streamflow for selected reach number from the output.rch file, and store it in each model output file, “stf_001.txt”, “stf_003.txt”, and “stf_068.txt”.
- `swatmf_pst_utils.stf_obd_to_ins()` creates an instruction file based on each stf_extracted files and match observed data in colname from the “stf_day.obd.csv” file, “stf_001.txt.ins”, “stf_003.txt.ins”, and “stf_068.txt.ins”.

To generate groundwater level simulation outputs, we first construct a DataFrame using grid ID and layer number information from the "modflow.obs" and "swatmf_out_MF_obs" files. This DataFrame is then used to create individual groundwater simulation output files. Finally, we create instruction files using the simulation output file names and corresponding observation column names.

```
>>> from swatmf.handler import SWATMFout
>>> m1 = SWATMFout(wd)
>>> df = m1.get_gw_sim()
>>> for col in df.columns:
        df.loc[:, col].to_csv(
          '{}.txt'.format(col), sep='\t', encoding='utf-8',
          index=True, header=False, float_format='%.7e'
          )

>>> swatmf_pst_utils.mf_obd_to_ins(
       simnam, colname,
       cal_start, cal_end
       )
```

- `simnam` is a list of groundwater simulation outputs, the dtw_sim249lyr2, 249lyr3, 1205lyr3, and 1203lyr3 files and `colname` "249lyr2", "249lyr3", "1205lyr3", "1203lyr3" in the “dtw_day.obd.csv” file.
- `swatmf_pst_utils.mf_obd_to_ins()` creates an instruction file based on each `simnam` files and match observed data in `colname` from the “dtw_day.obd.csv” file, “sim_g249lyr2.txt.ins”, “sim_g249lyr3.txt.ins”, “sim_g1203lyr3.txt.ins”, and “sim_g1205lyr3.txt.ins”.

6. Create PESTPP control file

The pyEMU library offers a streamlined approach to generating a PEST++ control file (control variable, stored in memory) by leveraging pre-established templates and instruction files.

```
>>> io_files = pyemu.helpers.parse_dir_for_io_files('.')
>>> pst = pyemu.Pst.from_io_files(*io_files)

Let’s write the control file based on your customization.
>>> pst.write(‘koki_zon.pst', version=2)
```

Users must customize initial parameter values, ranges, and group names. Additionally, default observation values should be replaced with actual data, and observation group names need to be adjusted. Detailed instructions can be found in the "01_zon_setup.ipynb" notebook within the repository.

7. Perform uncertainty analysis with running PESTPP-IES

First, we need to load the existing control file and add some PESTPP-IES specific options. As with most PEST++ options, these mostly have pretty decent default values; however, depending on your setup you may wish to change them. We highly recommend reading the PEST++ user manual for full descriptions of the options and their default values.

```
>>> pst_path = os.path.join(wd, ' koki_zon.pst ')
>>> pst = pyemu.Pst(pst_path)
>>> pst.pestpp_options["ies_num_reals"] = 300 # set the number of
ensemble!
>>> pst.control_data.noptmax = 10 # set the number of iterations

# update settings again and re-write the control file
>>> pst.write(os.path.join(wd, 'koki_zon_ies.pst'),version=2)
```

We need to specify the number which is adequate for your machine! Make sure to assign an appropriate value for the following num_workers variable.
You can check the number of physical cores avalable on your machine using psutils:

```
>>> import psutil
>>> num_workers = psutil.cpu_count(logical=False)
```

Next, we shall specify the PEST run-manager/master directory folder as m_d. This is where outcomes of the PEST run will be recorded. It should be different from the wd folder, which contains the "template" of the PEST dataset. This keeps everything separate and avoids silly mistakes.

```
>>> m_d = os.path.join(prj_dir, "koki_zon_ies")
```

The following cell deploys the PEST agents and manager and then starts the run using PESTPP-IES. Run it by pressing shift+enter. If you wish to see the outputs in real-time, switch over to the terminal window (the one which you used to launch the jupyter notebook). There you should see PESTPP-IES's progress. If you open the tutorial folder, you should also see a bunch of new folders there named worker_0, worker_1, etc. These are the agent folders. The `koki_zon_ies` folder is where the manager is running.

```
>>> os.chdir(prj_dir)
>>> pyemu.os_utils.start_workers(
        wd, # the folder which contains the "template" PEST dataset
        'pestpp-ies', #the PEST software version we want to run
        ' koki_zon_ies.pst', # the control file to use with PEST
        num_workers=num_workers, #how many agents to deploy
        worker_root='.', #where to deploy the agent directories; relative
to where python is running
        master_dir=m_d, #the manager directory
        )
```

8. Perform Sensitivity analysis with running PESTPP-SEN

Building upon the existing PEST++ interface established for uncertainty analysis, sensitivity analysis can be conducted by incorporating specific options and substituting the PESTPP-SEN utility for PESTPP. In this study, we applied the Morris screening method (elementary effects test) (Morris, 1991) to assess the relative influence of parameters on streamflow and groundwater head. This qualitative global sensitivity analysis (GSA) quantifies parameter sensitivity by calculating the change in model output (elementary effect) resulting from variations in a single parameter while holding others constant across a range of parameter values.

```
>>> pst_path = os.path.join(wd, 'koki_zon.pst')
>>> pst = pyemu.Pst(pst_path)
>>> pst.pestpp_options["gsa_method"] = “morris” # set GSA method
>>> pst.pestpp_options["gsa_morris_r"] = 10 # set sample size

# update settings again and re-write the control file
>>> pst.write(os.path.join(wd, 'koki_zon_morris.pst'),version=2)
>>> m_d = os.path.join(prj_dir, “koki_zon_morris”)
```

```
>>> os.chdir(prj_dir)
>>> pyemu.os_utils.start_workers(
        wd, # the folder which contains the "template" PEST dataset
        'pestpp-sen', #the PEST software version we want to run
        ' koki_zon_morris.pst', # the control file to use with PEST
        num_workers=num_workers, #how many agents to deploy
        worker_root='.', #where to deploy the agent directories; relative
to where python is running
        master_dir=m_d, #the manager directory
        )
```