PyWrfHydroCalib Manual: A Calibration toolkit of the WRF-Hydro Modeling System

Table of Content:

Introduction

WRF-Hydro, an open-source community model, is used for a range of projects, including flash flood prediction, regional hydroclimate impacts assessment, seasonal forecasting of water resources, and land-atmosphere coupling studies. The National Water Model (NWM) is a joint development effort between the National Center for Atmospheric Research (NCAR) and the National Weather Service (NWS) Office of Water Prediction (OWP). The NWM is a set of operational analysis and forecast configurations of the WRF-Hydro modeling system (Gochis et al., 2013) which runs over the entire continental United States (https://water.noaa.gov/about/nwm). Several upgrades to this modeling system have occurred in which calibration became increasingly a focus for model improvements. As such, the development of a robust workflow for calibrating parameters over individual basins occurred

with a focus on automating the process for hundreds of forecast points in the NWM domain. Initially, this workflow was developed for use on NCAR's Yellowstone HPC environment. However, recent advances have been made to make the calibration workflow more portable to other systems for individual use cases and testing. This document describes the various steps involved in setting up a WRF-Hydro calibration experiment, along with how to execute the calibration workflow and access the various results from it. Currently, this version of the calibration workflow has been designed to calibrate against observed streamflow. However, future iterations will expand the focus to calibrate against other observed hydrologic states, such as snow, inundation, and soil moisture.

Prerequisites

- Installation of the WRF-Hydro/NWM modeling system (use a version that is pointed out in the release notes of the package)
- Preparation of a modeling domain with a forecast streamflow point
- Preparation of associated forcing files necessary to run the model simulations
- Preparation of input observation files that will be used in the calibration workflow

Calibration is an advanced topic that goes beyond setting up the NWM or WRF-Hydro for a model simulation. Some understanding of the preparation of datasets, hydrologic parameters, and their associated impact on model states is needed. You will decide which parameters to calibrate, along with the proper ranges, domain size, and the length of the calibration simulation/evaluation period. It is highly encouraged that you run sensitivity analysis for the modeling domain to explore which parameters have significant impact on the modeled states being calibrated.

The model must be compiled and built before the calibration workflow can be set up. For in-depth information on installation of the model, see the WRF-Hydro Technical Description and User Guide available from the WRF-Hydro Modeling System website https://ral.ucar.edu/projects/wrf_hydro. Some library dependencies for R and Python will be met during the process of installing the model, namely the NetCDF libraries.

In addition to compiling the WRF-Hydro code, there are a set of parameter NetCDF files that are generated from the GIS pre-processor (see WRF-Hydro user guides and documentation) that are modified during the calibration workflow. The following parameter files from the output of the WRF-Hydro GIS Preprocessing tool, which are unique to each modeling domain, are needed in addition to the regular input files for the calibration:

- **soil_properties.nc** - A 2D soil property file that contains values impacting sub-surface hydrologic response.
- **Fulldom.nc** - The 2D geospatial fabric utilized for the high-resolution routing.
- **GWBUCKPARM.nc** - The 2D NetCDF groundwater bucket parameter file.
- **HYDRO_TBL_2D.nc** - The 2D NetCDF of surface hydrologic parameters impacting hydrologic response for overland flow routing.
- **CHANPARM.TBL** - The table file containing the channel properties in case the user is calibrating the channel properties using the CHANPARM.TBL

Calibration Requirements

The following steps are involved in setting up the calibration workflow:

- Installation of all technical dependencies for the workflow These include:
    - Installation of R and associated packages.
        - data.table
        - ggplot2
        - ncdf4
        - plyr
        - gridExtra
        - hydroGOF
        - qmap
        - zoo
        - raster
        - lubridate
        - reshape2
    - Installation of Python and associated packages.
        - netCDF4
        - pandas
        - numpy
        - psycopg2
        - psutil
        - xarray

The calibration workflow relies on sqlite database tables heavily for storage of metadata information associated with the modeling domains, statuses on various workflow components, calibrated parameter values, and output statistics.

Calibration Workflow & Program Description

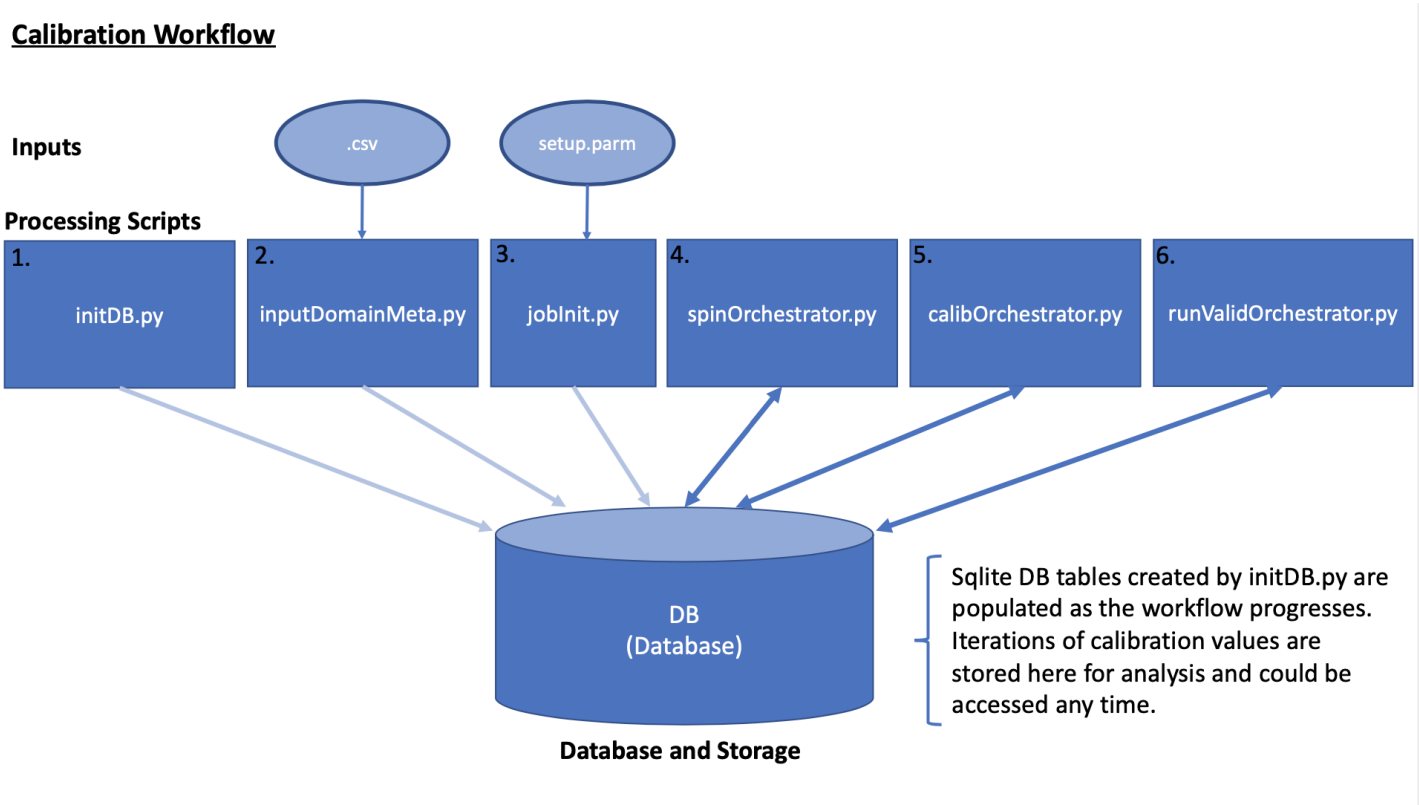

Figure 1. Following are descriptions of the high-level programs that have been developed for running the calibration workflow.

Programs:

initDB.py: This program is used one time to initialize the calibration database and associated tables used during the experiment. The upcoming section will describe the database and associated tables in more detail.

inputDomainMeta.py: This program reads in a CSV file you will need to fill out that describes modeling domains to be used for calibration. This information is entered into the database for later workflow use. More description on the CSV will occur during the setup section.

jobInit.py: This program is run to establish a calibration 'experiment'. The program reads a config file (explained later in the setup section) and sets up the necessary run directories, paths to necessary files, and inputs associated metadata into the database. Upon successful completion, the program will return a unique job ID value which you will use in subsequent programs to run the calibration.

spinupOrchestrator.py: This is the first program that is run to initialize the calibration experiment. The only mandatory argument to this program is the unique job ID for the calibration experiment. The main purpose of this program is to run the NWM/WRF-Hydro spinup for all domains being calibrated. This program needs to successfully complete before moving onto the next step.

calibOrchestrator.py: This is the second program that is run in the calibration workflow. As with spinup.py, the only mandatory argument is the job ID value. This program runs the main workflow to

adjust parameter values, execute interim model simulations, evaluate model output against observations, and further adjust parameter values. This program must be completed successfully before moving onto the next step.

runValidationOrchestrator.py: This is the third and final main program in the calibration workflow. The only mandatory argument is the unique job ID value associated with the calibration experiment. This program manages running the model with the final calibrated parameters over a specified evaluation period for the evaluation of the parameters.

Overview of the SQL Database

Upon completion of the jobInit.py program, a database called 'wrfHydroCalib_DB' (or a different name specified by the user) is established in SQL. It contains several tables that can be utilized by you as well as in the workflow. In addition to the database being established, a 'user' called 'WH_Calib_rw' is created for access to the database. The following empty tables are created in the NWM_Calib_DB database:

- *Domain_Meta*
- *Job_Meta*
- *Job_Params*
- *Calib_Params*
- *Calib_Stats*
- *Valid_Stats*

Domain_Meta: is a table that contains metadata about each basin/domain you are using in your calibration efforts. There is no limit to how many 'domains' you can enter in here as it depends on the scope of the experiment. Information is entered into this table by the *inputDomainMeta.py* program. The following table columns exist in Domain_Meta:

domainID	A unique integer ID associated with a particular domain in the table. This value is automatically generated as the information is entered into the database.
gage_id	A character entry that describes the ID associated with whatever stream gauge the user is calibrating against. For example, a USGS ID, CA DWR ID, etc.

link_id	A unique integer feature_id value that associates an unique point in the streamflow output files to the stream gauge being used for calibration. The user will need to determine which feature_id in their output files corresponds to the observation point being calibrated.
domain_path	A character entry that points to a directory containing all necessary input domain files for this associated domain (i.e. geogrid file, wrfinput file, etc). It is assumed that a 'FORCING' subdirectory will be contained within this path that contains necessary input forcing files. The user can also create a symbolic link to the actual forcings directory as well. There should also be an OBS directory that has the obsStrData.Rdata file containing the streamflow observations.

gage_agency	A character entry that describes the agency in charge of the reporting stream gauge (i.e. USGS, CA DWR, etc).
geo_e (NWM only)	An integer entry describing the cutout from the parent NWM domain. Specifically, the NWM conus 1 km domain row that specifies the eastern edge of your cutout domain. If the domain is not a NWM cutout, this value will be -9999.
geo_w (NWM only)	An integer entry describing the cutout from the parent NWM domain. Specifically, the NWM conus 1 km domain row that specifies the western edge of your cutout domain. If the domain is not a NWM cutout, this value will be -9999.
geo_s (NWM only)	An integer entry describing the cutout from the parent NWM domain. Specifically, the NWM conus 1 km domain row that specifies the southern edge of your cutout domain. If the domain is not a NWM cutout, this value will be -9999.
geo_n (NWM only)	An integer entry describing the cutout from the parent NWM domain. Specifically, the NWM conus 1 km domain row that specifies the northern edge of your cutout domain. If the domain is not a NWM cutout, this value will be -9999.
hyd_e (NWM only)	An integer entry describing the cutout from the parent NWM domain. Specifically, the NWM conus 250 meter domain column that specifies the eastern edge of your cutout domain. If the domain is not a NWM cutout, this value will be -9999.
hyd_w (NWM only)	An integer entry describing the cutout from the parent NWM domain. Specifically, the NWM conus 250 meter domain column that specifies the western edge of your cutout domain. If the domain is not a NWM cutout, this value will be -9999.
hyd_s (NWM only)	An integer entry describing the cutout from the parent NWM domain. Specifically, the NWM conus 250 meter domain row that specifies the southern edge of your cutout domain. If the domain is not a NWM cutout, this value will be -9999.
hyd_n (NWM only)	An integer entry describing the cutout from the parent NWM domain. Specifically, the NWM conus 250 meter domain row that specifies the northern edge of your cutout domain. If the domain is not a NWM cutout, this value will be -9999.
geo_file	A character entry describing the path to the geogrid file necessary to run the model
land_spatial_meta_file	A character entry describing the path to the optional 2D NetCDF file created by the WRF-Hydro GIS pre-processing for creating CF-compliant land surface output. This entry will be -9999 if it's missing.
wrfinput_file	A character entry describing the path to the wrfinput file necessary to run the model
soil_file	A character entry describing the path to the soil_properties.nc file necessary to run the model. This is a key 2D file containing parameters that are adjusted during the calibration process.
fulldom_file	A character entry describing the path to the Fulldom.nc file necessary to run the model. This is a key 2D file containing parameters that are adjusted during the calibration process.
rtlink_file (NWM only)	A character entry describing the path to the route link file necessary to run the model. For non-NWM calibrations, this entry will be -9999.

spweight_file (NWM only)	A character entry describing the path to the spatial weight file necessary to run the model. For non-NWM calibrations, this entry will be -9999.
gw_file	A character entry describing the path to the groundwater bucket parameter file necessary to run the model. This is a key parameter file that gets adjusted during the calibration process.
gw_mask	A character entry describing the path to the groundwater mask file used for groundwater configuration. If not used, this entry will be -9999.
lake_file	A character entry describing the path to the lake parameter file necessary to run lakes within the model. If your domain does not contain lakes, this value will be set to -9999.
forcing_dir	A character entry describing the path to the directory containing necessary forcing files to run the model
obs_file	A character entry describing the path to the pre-processed R file containing the observed streamflow
site_name	A character entry giving a description of the stream gage
lat	The floating point latitude of the location of the stream gage
lon	The floating point longitude of the location of the stream gage
area_sqmi	The area of the watershed being calibrated in square miles
area_sqkm	The area of the watershed being calibrated in squared km
county-cd	An integer entry describing the county code for where the stream gage resides
state	The state the stream gage resides within
huc2	The HUC2 that this basin falls within
huc4	The HUC4 that the basin falls within
huc6	The HUC6 that the basin falls within
huc8	The HUC8 that the basin falls within
ecol3	The ecological level 3 region the basin falls within
ecol4	The ecological level 4 region the basin falls within
rfc	The NWS River Forecast Center the basin falls within
dx_hydro	The grid spacing on the routing grid. This is calculated by the workflow.
agg_factor	The aggregation factor between the routing and land surface grid. This is calculated by the workflow.

Job_Meta is a table that contains metadata about the calibration experiment being run. This information is entered into the table when jobInit.py is successfully run to completion. Key table variables (calib_complete, valid_complete, su_complete) are updated throughout the calibration workflow as specific tasks are completed. Additionally, other table variables from this table are used during the workflow to create necessary namelist files and symbolic links to the necessary input files to execute the model. Note that there are a few more columns which are related to the sensitivity analysis

available in the package, slack messaging and job submission properties that are not explained here. The following table columns exist in the table. Note the below is not a complete list, and there are a few more columns that are not mentioned in the below table.

jobID	A unique integer value associated with the calibration experiment. This value is selected by the user at the time of the experiment initiation.
Job_Directory	A character entry describing the top-level directory containing all output for your calibration experiment. Each basin will be a sub-directory under this top-level directory. The jobInit program will create and populate sub-directories appropriately.
calib_flag	Flag indicating if this is a calibration experiment. Flag = 1 indicates this is an calibration experiment.
calib_table	Path to where the calibration table containing the parameters to be calibrated with their default, min and max values exists.
date_su_start	A datetime entry that specifies the start of the model spinup period
date_su_end	A datetime entry that specifies the end of the model spinup period
su_complete	A 0/1 integer entry indicating if the spinup has completed for all basins in the experiment
date_calib_start	A datetime entry that specifies the start of the calibration period
date_calib_end	A datetime entry that specifies the end of the calibration period
date_calib_start_eval	The beginning date within the calibration period that will be used to perform analysis against observations for parameter adjustment. The analysis period will run until the end of the calibration period
num_iter	An integer entry indicating the number of model iterations to take place for calibrations
calib_complete	A 0/1 integer entry indicating if the calibration has completed for all basins in the experiment
valid_start_date	A datetime entry that specifies the start date for the validation simulation period
valid_end_date	A datetime entry that specifies the end date for the validation simulation period
valid_start_date_eval	A datetime entry that specifies the beginning date within the validation period to perform analysis on. Analysis will take place from this date until the end of the validation simulation period.
valid_complete	A 0/1 integer entry indicating if the validation has completed within all basins in the calibration experiment
acct_key	An optional character string indicating an account key to run the jobs on if you are running with BSUB/QSUB/Slurm on an HPC environment

que_name	An optional character string directing the workflow which que to place model simulations into when submitting jobs.
num_cores_model	An integer entry indicating the number of CPU cores to execute the model over
num_nodes_model	An integer entry indicating the number of compute nodes running the model simulations over. If you are using mpiexec/mpirun, set this to 1 in the configuration file.
num_cores_per_node	Number of available cores per each node.
job_run_type	An integer entry indicating the method of executing the model simulations
exe	A character entry pointing to the compiled NWM/WRF-Hydro executable to run the simulations
num_gages	An integer entry indicating the total number of basins in the experiment
owner	The local owner on the computer running the simulations. This is established during the jobinit.py phase.
email	An email to pipe status/error messages to during the calibration workflow

Job_Params is a table describing the various parameters chosen for calibration for this particular experiment across all the basins. The following table columns exist in the table:

jobID	An integer entry connecting the table to the unique job ID created during initialization of your experiment
param	A character entry describing the name of the parameter value being calibrated (i.e. ‘bexp’,’refkdt’,etc)
defaultValue	Default value used to initialize the parameter of interest for the first model iteration
min	The minimum possible parameter value that can be searched for during the calibration workflow
max	The maximum possible parameter value that can be searched for during the calibration workflow

Calib_Params is a dynamic table updated as the calibration workflow works through the model iterations. The table describes the parameter values calculated after every model iteration and stores them in this table. The following table columns exist:

jobID	An integer entry connecting the table to the unique job ID created during the initialization of your experiment
domainID	An integer entry connecting the basin of interest to the entries in the Domain_Meta table describing each basin being calibrated

iteration	An integer entry describing which iteration the parameter values represent
paramName	Character entry indicating the name of the parameter (i.e. ‘bexp’,’refkdt’,etc)
paramValue	The parameter value calculated for this particular model iteration

Calib_Stats is a dynamic table updated as the calibration workflow works through the model iterations. The table describes the analysis statistics for each model iteration, along with a status value that aids the workflow in monitoring jobs. The following table columns exist in the Calib_Stats. For more information on the formulation of each error statistics refer to the code (calib_workflow.R and calib_utils.R).

jobID	An integer entry connecting the table to the unique job ID created during the initialization of your experiment
domainID	An integer entry connecting the basin of interest to the entries in the Domain_Meta table describing each basin being calibrated
iteration	An integer entry describing which iteration the parameter values represent
objfnVal	The value of the objective function calculated after an iteration completes. The user will choose which error metric to use as the objective function during the initialization of the experiment.
bias	The bias against observations calculated after an iteration completes
rmse	The root-mean-squared error calculated after an iteration completes
cor	The correlation calculated after an iteration completes
nse	The Nash-Sutcliff value calculated after an iteration completes
nselog	The log of the Nash-Sutcliff calculated after an iteration completes
nnse	NNSE = 1 / (2 -NSE)
nnsesq	NNSESQ = 1 / (2 - NSE^2)
kge	The kge value calculated after an iteration completes
fdcerr	The flow-duration curve error value calculated after an iteration completes
msof	The msof error value calculated after each iteration
hyperResMultiObj	Maximize NNSE while minimizing flow volume and peak bias error
lbem	Improved theoretical estimator of efficiency based on NSE for when hydrologic data is skewed (non-normal) and exhibits periodicity (e.g., seasonality), which is often the case for daily and subdaily streamflow.
lbemprime	Improved theoretical estimator of efficiency based on KGE for when hydrologic data is skewed (non-normal) and exhibits periodicity (e.g., seasonality), which is often the case for daily and subdaily streamflow. (Lamontagne 2020 https://doi.org/10.1029/2020WR027101)
corr1	Stedinger’s (1981) lognormal estimator of correlation (corr1) Purpose: Improved

	estimator of correlation when hydrologic data is skewed (non-normal), which is often the case for daily and subdaily streamflow https://doi.org/10.1080/02626667.2019.1686639
pod	Probability that a flood was observed when it was forecasted; where 1 is a perfect score and 0 is the worst.
far	Probability that a flood was forecasted, but not observed; where 0 is perfect and 1 is the worst
csi	The proportion of correctly forecast floods over all floods, either forecast or observed
eventmultiobj	Weighted peak bias and volumn bias over the events
peak_bias	Streamflow peak bias over the events
peak_tm_err_hr	Streamflow peak timing error over the events
event_volume_bias	Streamflow volumn bias over the events
cor_snow	Correlation coefficient for snow
rmse_snow	RMSE of snow
bias_snow	Bias of snow
nse_snow	NSE of snow
kge_snow	KGE of snow
cor_soil	Correlation coefficient of soil moisture
rmse_soil	RMSE of soil moisture
bias_soil	Bias of soil moisture
nse_soil	NSE of soil moisture
kge_soil	KGE of soil moisture
kge_alpha_soil	KGE alpha for soil moisture
best	A 0/1 entry that indicates if this iteration produced the best results based off the minimization of the objective function value
complete	A dynamic floating point value that specifies how complete (or incomplete) this calibration iteration is

Valid_Stats is a table that describes the error metrics associated with both the default parameter values chosen at the beginning of the experiment, along with the final calibrated values. The following table columns exist in the Valid_Stats. In addition to the ones listed below, all the error metrics listed in the Calib_Stats table also exist in the Valid_Stats which is not shown to reduce repetition.

jobID	An integer entry connecting the table to the unique job ID created during the initialization of your experiment

domainID	An integer entry connecting the basin of interest to the entries in the Domain_Meta table describing each basin being calibrated
simulation	A character entry describing if these error metrics represent a simulation that used either default or calibrated parameter values
evalPeriod	A character entry describing which simulation period was used for calculation of error metrics (i.e. calib,valid, full)
objfnVal	The value of the objective function calculated from this model period

Database Setup

Once you have the necessary library dependencies installed, you have some modeling domains setup for the model, and you have a compiled executable for NWM/WRF-Hydro, you are ready to begin setting up your calibration workflow. First thing in the chain of the commands would be to initialize the database. To create this, you must execute the initDB.py program. Run the following commands:

```
python PATH/TO/PyWrfHydroCalib/initDB.py --optDbPath
PATH/TO/DATABASE.db
```

This command will run the initDB.py script and initialize an empty database. The location and the name of the database is defined after the user --optDbPath. So the name of the database in the above command would be (DATABASE.db).

Remember you only need to execute this program once. If you try to run it again, you will receive an error indicating you have already created the database:

"ERROR: PATH/TO/DATABASE.db Already Exists. "

Entering Basin Information into the Database

First, you will need to fill out a .csv file that will contain information on each of your domains. To locate the .csv file, navigate to the calibration code directory and list the subdirectories. The subdirectory called 'setup_files' contains a .csv file called 'domainMetaTemplate.csv'. Open this file up for editing. It contains the following column headers:

- site_no
- link
- hyd_w (NWM only) (Optional)
- hyd_e (NWM only) (Optional)
- hyd_s (NWM only) (Optional)
- hyd_n (NWM only) (Optional)
- geo_w (NWM only) (Optional)
- geo_e (NWM only) (Optional)
- geo_s (NWM only) (Optional)
- geo_n (NWM only) (Optional)
- dirname

- agency_cd
- site_name
- lat (Optional)
- lon (Optional)
- area_sqmi (Optional)
- area_sqkm (Optional)
- county_cd (Optional)
- state (Optional)
- HUC2 (Optional)
- HUC4 (Optional)
- HUC6 (Optional)
- HUC8 (Optional)
- ecol3 (Optional)
- ecol4 (Optional)
- rfc (Optional)

Beneath this header, you will see a sample entry for one basin with an associated entry for each column header. It is recommended that you copy this file and edit it based on the sample included with the code. If you have multiple basins you would like to calibrate, you will need to make multiple entries into the CSV file. Alternatively, you could enter each basin one CSV at a time as you gather information.

While the list above contains a large variety in potential metadata for each basin, it should be noted that not all fields need to be known to enter this information into the database. **The fields that are of most importance, and should be filled are site_no, link, and dirname.**

If you do not know the remaining metadata fields, you can simply leave the values as -9999, or 'nan' in the template. Having extensive metadata information in the database allows for versatile searches to create a multitude of various experiments if you have a significant number of areas you are interested in calibrating.

The 'dirname' field is crucial as this is where all your domain files associated with your basin are located (geogrid, Fulldom, etc). In addition, within that directory, it is expected that a **"FORCING"** subdirectory is placed there containing all the necessary forcing files, or symbolic links to those forcing files. There should also be an **OBS** directory that has the **obsStrData.Rdata** file containing the streamflow observations with the following format: Note, the basinTypeName and basinType are not required, if the basinType in specified in the setup file.

```
site_no  POSIXct              obs         threshold   basinTypeName basinType
02465493 2013-09-01 05:00:00  0.5097032    2.9715       regular      3
02465493 2013-09-01 06:00:00  0.5097032    2.9715       regular      3
02465493 2013-09-01 07:00:00  0.5097032    2.9715       regular      3
```

Where the obs is the streamflow values in cms. Once you are comfortable with the information you have entered into the .csv file, you are now ready to enter this information into the database (Domain_Meta). You need to run the program 'inputDomainMeta.py' with the path to the CSV file as an argument and the path to the database as the second argument. For example:

```
python                          PATH/TO/PyWrfHydroCalib/inputDomainMeta.py
```

```
PATH/TO/domainMeta.csv --optDbPath PATH/TO/DATABASE.db
```

This Will enter information from the CSV file into the database table Domain_Meta. If you could check the content of the sqlite database display the Domain_Meta table.

Potential errors may arise if:

- You did not enter the correct number of columns into the .csv file
- The directory you entered for the input domain files does not exist.
- Expected files within the directory (FORCING subdirectory, geogrid, Fulldom, etc.) do not exist.
- The headers in the .csv file are not the expected format the program is expecting, or contain the incorrect header column names. This is why it is recommended to simply make a copy of the template file included and edit it appropriately for your basins.

Additionally, you may receive warning messages if certain optional files are not found. For example, the workflow will look for a lake parameter file. However, it is possible your model domain contains no lakes, so this file is unnecessary. The workflow will provide a warning message indicating this file was not found. Once this step is complete for all the basins you plan on calibrating the model to, you are now ready to create your configuration file, 'setup.parm' and initialize your experiment.

NOTE: if you are calibrating snow or soil moisture, then there should be `obsSnowData.Rdata` and `obsSoilData.Rdata` in the OBS folder, respectively.

Initializing Your Experiment

Creating the setup.parm File

The primary file you will be editing in preparation for setting up a calibration workflow job is the 'setup.parm' file. It is best to think of this file as a master configuration file to guide the workflow. A template file is located under /setup_files/setup.parm in the PyNWMCalib code repository. This file contains multiple options that define how the workflow will submit jobs for models/analysis, which basins to calibrate from the database, methods for reporting errors to the user, model physics options, and paths to general parameter files and executables.

The 'setup.parm' file is divided up into sections: logistics, gageInfo, lsmPhysics, forcing, modelTime, hydroIO, and hydroPhysics. The first section of the setup.parm file is 'logistics', which guides the workflow. You can find a template setup.parm file in /setup_files/setup.parm. The following fields need to be filled in:

logistics:

outDir	The top-level directory where all output from the workflow will be placed. Subdirectories for each basin will be created under this directory. (Make sure you have write permission here)
expName	A unique name you would like to give your calibration experiment

acctKey (optional)	The account key to run your jobs under if you are running the workflow with BSUB, QSUB, or Slurm. **Leave blank if you do not need an account key on your systems**.
optQueNameModel	This is an optional entry for BSUB, QSUB, and Slurm job submissions for model simulations. If your system requires a que to place your jobs into, you can fill out this field appropriately. **Otherwise, you can leave this blank if you do not need a que system.**
nCoresModel	The number of CPU cores you would like to use to run NWM/WRF-Hydro.
nNodesModel	This is the number of compute nodes you plan on running over for your model simulations. For example, if you plan on running your model domain over 100 CPUs, but your HPC environment has 20 CPUs per node, you may want to set this value to 5. If you are running with mpiexec/mpirun, you can leave this value as 1.
nCoresPerNode	The number of available cores per node on your HPC system.
runSens	Flag to turn sensitivity analysis on: 0 - Off, 1 - On. (NOT required for calibration, leave it 0)
sensParmTbl	Path to the sensitivity parameter table (NOT required for calibration)
runCalib	Flag to turn calibration on: 0 - Off, 1 - On. (Set it to 1 for calibration experiment and 0 for sensitivity analysis)
calibParmTbl	Path to the calibration parameter table
runTroute	Leave it at 0, this is not a supported activity.
trouteConfig	Leave it as an empty string, not used.
moduleLoadStr	List of the modules/libraries that the user would like to load, for example the python environment, R environment etc.
dailyStats	Model is usually run at the hourly time scale. This flag specify whether the calculation should be done at the hour or daily time scale. (0 - Run hourly stats, 1 - Run daily stats)
dbBackup	Flag to turn on/off database backup. If on, the database file will be locked and backed up once an hour during the execution to the job directory output file.
coldStart	This is a flag for the user to bypass the spinup and run calibrations/validations/sensitivity analysis from cold starts. Note: This is highly discouraged as a spinup allows for stable hydrologic states: 0 - Off, 1 - On.
optSpinFlag	Optional spinup flag for substituting a both land and hydro restart files in place of a spinup. NOTE: The user must provide BOTH a hydro and land spinup state in the basin domain directories for ALL basins being used in this experiment. Expected file naming conventions for the expected restart files are as follows: ● Land: LandRestartSubstitute.nc ● Hydro: HydroRestartSubstitute.nc

	(0 = No substitute spinup files, 1 = Use optional spinup files)
stripCalibOutputs	Additional new flags to allow the user to only run the model with extremely minimal outputs (Monthly restart files, no streamflow output, and monthly land output files.) This allows for the model to minimize I/O time during the initial period of the calibration simulations where no evaluation is taking place. Flag to turn option on (1) or off (0)
stripCalibHours	Specify the initial time period (hours) for each simulation to contain minimal output.
jobRunType	The method of running your job submissions. ○ 1 - Launch via BSUB ○ 2 - Launch via QSUB ○ 3 - Launch via slurm ○ 4 - No job scheduler. Run via MPI
mpiCmd	Specify the MPI command to use (e.g. mpiexec -np)
cpuPinCmd	Specify the CPU pinning command to supplement the MPI command to pin the model to specific CPUs.
numIter	The number of simulation iterations you would like to have for your calibration experiment
calibMethod	The algorithm used for searching the parameter space. Right now only 'DDS' is accepted. Future iterations of the workflow will expand this to include additional algorithms, such as SCE.
enableStreamflowCalib	1 means calibrating streamflow and 0 means not calibrating streamflow
enableSnowCalib	1 means calibrating snow and 0 means not calibrating snow
enableSoilMoistureCalib	1 means calibrating soil moisture and 0 means not calibrating soil moisture
streamflowObjectiveFunction	Objective function to be used for streamflow
snowObjectiveFunction	Objective function to be used for snow
soilMoistureObjectiveFunction	Objective function to be used for soil moisture
streamflowWeight	Weight given to the streamflow objective function in the total objective function. The total objective function is a weight value of all the objective functions.
snowWeight	Weight given to the snow objective function

soilMoistureWeight	Weight given to the soil moisture objective function
basinType = NA weight1Event = 0.6 weight2Event = 0.4	# basinType has flags: 0: snowy; 1: slow; 2: flashy # weight1Event and weight2Event are weights for peak bias and volume bias to get combined metric
enableMask	● Specify whether to use the mask to mask out some part of the basins from calibrating or not ● if enableMask set to 0, no need to provide a mask ● if enableMask set to 1, then a mask should be provided on the coarse grid, in tif format ● The mask should be set to 1 where we want to keep the parameters as is and set to 0 where we would like to calibrate the parameters
enableMultiSites	● Specify whether to calibrate to more than one streamflow gages or not ● if enableMultiSite set to 1, a file containing the list of the gages of interest and the corresponding weights in the calculation of the objective function is required. Name of the file is hardcoded to "calib_sites.csv" that is placed under the domain dir and contains the following fields: ○ FID : Feature id of the gage as it appears in the CHANOBS ○ site_no : site identifier as it appears in the obsStrDate.Rdata file ○ weight : weight to be given to site when calculating the objective function
ddsR	The DDS search radius. Default is 0.2. It's recommended to leave this value as 0.2.
email	Enter the email you would like to have status updates and error messages sent to. If you leave this blank, or place a missing value in there 'MISSING', the workflow will simply print error messages out to the screen as opposed to sending an error message via email.
wrfExe	The path to the NWM/WRF-Hydro executable you would like to use for this calibration experiment. **The user will need to compile and build an executable on their system prior to running the calibration workflow.**
genParmTbl	The path to the GENPARM.TBL to use.
mpParmTbl	The path to the MPTABLE.TBL to use
urbParmTbl	The path to the URBPARM.TBL to use
vegParmTbl	The path to use the VEGPARM.TBL to use
chanParmTbl	The path to use the CHANPARM.TBL to use
soilParmTbl	The path to the SOILPARM.TBL to use
bSpinDate	The beginning date for the spinup (YYYY-MM-DD)

eSpinDate	The ending date for the spinup (YYYY-MM-DD)
bCalibDate	The beginning date for the calibration period (YYYY-MM-DD)
eCalibDate	The ending date for the calibration period (YYYY-MM-DD)
bCalibEvalDate	The beginning date for evaluation in the calibration (YYYY-MM-DD)
bValidDate	The beginning date for the validation period (YYYY-MM-DD)
eValidDate	The ending date for the validation period (YYYY-MM-DD)
bValidEvalDate	The beginning date for evaluation in the validation period (YYYY-MM-DD)

It is important to note that genParmTbl, mpParmTbl, urbParmTbl, vegParmTbl, chanParmTbl, and soilParmTbl files are global parameter tables applied to all of your basins. Note that these tables are dependent on the WRFHydro version and are distributed with the code, and users must use the right TBL files.

gageInfo:

The next section 'gageInfo' tells the workflow which basins the model is going to be calibrating over.
gageListSQL: The SQL command to use to pull basins for calibration from the Domain_Meta table. If you are using the gageListFile option, please leave this field blank.

- Example - select * from 'Domain_Meta' where state=='California'; will pull all basins from Domain_Meta that are tagged as being in California.

gageListFile: A list of gages to pull from. This can be used in place of an SQL command. Please leave the gageListSQL field blank if using this option.A sample gage list file is provided in /setup_files/gage_list_template.csv.

lsmPhysics:

The next section 'lsmPhysics' tells the workflow which land surface physics options (as entered into the namelist.hrldas file) to use for this calibration experiment.

dynVegOption	DYNAMIC_VEG_OPTION
canStomResOption	CANOPY_STOMATAL_RESISTANCE_OPTION
btrOption	BTR_OPTION
runoffOption	RUNOFF_OPTION
sfcDragOption	SURFACE_DRAG_OPTION
frzSoilOption	FROZEN_SOIL_OPTION
supCoolOption	SUPERCOOLED_WATER_OPTION
radTransferOption	RADIATIVE_TRANSFER_OPTION
snAlbOption	SNOW_ALBEDO_OPTION

pcpPartOption	PCP_PARTITION_OPTION
tbotOption	TBOT_OPTION
tempTimeSchOption	TEMP_TIME_SCHEME_OPTION
sfcResOption	SURFACE_RESISTANCE_OPTION
glacierOption	GLACIER_OPTION
soilThick	The thicknesses of the soil layers in NoahMP (meters, four layers)
zLvl	Level of winds in the forcings for downscaling purposes (meters)

forcing:

The next section 'forcing' tells the workflow the nature of the forcings you are using to force the simulations with.

forceType	The forcing type specified in the namelist.hrldas file.

modelTime:

The next section 'modelTime' tells the workflow the time stepping of the model and its associated outputs. While for many outputs, the output time step may not be important for parameter estimation, you should set the 'hydroOutDt' to a frequency high enough that the calibration code will capture daily, if not hourly streamflow patterns.

forceDt	The input forcing timestep (seconds)
lsmDt	The NoahMP model timestep (seconds)
lsmOutDt	The NoahMP output timestep (seconds)
lsmRstFreq	The NoahMP restart file frequency (seconds)
hydroRstFreq	The output frequency (seconds) of the hydro restart files
hydroOutDt	The output frequency (seconds) of the hydro output files. Keep these either hourly or daily for proper calibration.

hydroIO:

The next section 'hydroIO' contains options for controlling the hydro output files.

rstType	Flag (0/1) for overwriting soil variables from the routing restart file
ioConfigOutputs	Flag specifying different types of output variables to produce. All options will produce streamflow in your output files. Please see WRF-Hydro documentation for a more detailed description of this option.

ioFormOutputs	Flag specifying the level of internal compression to utilize with NetCDF4 libraries
chrtoutDomain	the CHRTOUT_DOMAIN output flag
chanObsDomain	The CHANOBS_DOMAIN output flag. It's recommended to set this to 1 for streamflow calibration
chrtoutGrid	CHRTOUT_GRID flag to turn out gridded routing output
lsmDomain	Flag for turning on/off outputs from the LSM states in the hydro model
rtoutDomain	Flag for turning on/off high-resolution routing gridded output
gwOut	Flag for turning groundwater output on/off
lakeOut	Flag for turning lake output on/off
frxstOut	Flag for turning on the optional text file output for streamflow
resetHydroAcc	Flag for resetting accumulated variables contained in the hydro restart file
streamOrderOut	Flag for specifying the strahler order for streamflow output

hydroPhysics:

The last section "hydroPhysics' specifies physics flags specific to the hydro physics in NWM/WRF-Hydro

dtChSec	Channel routing timestep (seconds)
dtTerSec	Surface/Subsurface routine timestep (seconds)
subRouting	On/off switch for subsurface routing
ovrRouting	On/off switch for overland flow routing
channelRouting	On/off switch for channel routing
rtOpt	Overland routing option
imperv_adj	Specify whether to adjust overland flow parameters based on imperviousness
chanRtOpt	Channel routing option
udmpOpt	User-defined mapping option. For non-NWM configurations, this is 0.
gwBaseSw	Groundwater bucket option
gwRestart	Initialization flag for the groundwater bucket
enableCompoundChannel	Specify whether to use compound channels. This is for NWM ONLY.
compoundChannel	Activate the compound channels.
enableGwBucketLoss	Specify whether to enable the groundwater bucket loss function in the hydro.namelist file.

bucket_loss	Activate the groundwater bucket loss function.

Creating the calib_parms.tbl File

In addition to the 'setup.parm' file, the 'calib_parms.tbl' file is needed to direct the workflow to determine which model parameters will be calibrated, along with the range of parameter values. A template table is located under /setup_files/calib_parms.tbl which you can copy and edit for your own calibration workflow experiment.

Within this table, you will find all the potential parameters to calibrate, along with a 'calib_flag' of 1 or 0. This flag will turn calibration on (1) for that parameter or off (0). The 'minValues' and 'maxValues' specify the range of potential parameter values to calibrate over. This template has a broad range of values. The 'ini' column specifies the default values to be used for either default un-calibrated values, or the initial values going into the calibration workflow. It is up to you to determine what range is best for your calibration experiment. It is highly encouraged to perform a sensitivity analysis over your region of interest to help determine which parameters have significant impact on hydrologic response.

Once you are satisfied with the 'setup.parm' file and 'calib_parms.tbl', you are ready to initialize your experiment using 'jobInit.py'. This program will use the parameter table, along with the setup.parm file and the specified job ID by user. Enter the following command:

```
python      PATH/TO/PyWrfHydroCalib/jobInit.py      PATH/TO/setup.parm
--optExpID JobID --optDbPath PATH/TO/DATABASE.db
```

JobID needs to be an integer number. The program does some broad checking of options entered into the 'setup.parm' file to make sure they make reasonable sense before proceeding. However, it is up to you to ensure you are choosing the right modeling options for your experiment. Once the program has completed successfully, it will return the newly created job ID value to you.

```
WORKFLOW HAS BEEN SETUP FOR OWNER: X JOB ID = JobID
```

X above is the name of the user running the experiment (extracted from the provided email address) and JobID is the integer number provided by the user. Remember this job ID value as you will use it to execute subsequent programs in the workflow. Also, you will notice that within the top level output directory specified in the 'setup.parm' file, there are now subdirectories for each basin being calibrated. They are named according to the gage ID values entered into the 'Domain_Meta' table. Within each basin subdirectory, there are further subdirectories for the various components of the workflow.

- **FORCING** - A symbolic link to the forcing directory for this particular basin
- **OBS** - The directory containing symbolic links to the observation files necessary for calibration
- **RUN.CALIB** - The directory that contains output for the calibration iterations.
- **RUN.SPINUP** - The directory that contains output for the calibration pinup

- **RUN.VALID** - The directory that contains output for the calibration validation

You may also notice that in the top level output directory containing all the basins, there is a copy of the setup.parm file. **Please do not modify or remove this file** as it is used by the calibration workflow for reference. Also, it can be used to obtain the job ID for your calibration experiment if you forget it (see Utilities section).

Workflow Spinup

The next logical step in the calibration workflow is to spinup the model for each basin in your experiment. The following command will run your spinup for all basins in this experiment. Navigate to the calibration python code directory and run the following:

```
python PATH/TO/PyWrfHydroCalib/spinOrchestrator.py JobID --optDbPath
PATH/TO/DATABASE.db
```

Note: `JobID` is the unique job ID value created when you initialized your experiment.

The workflow will then proceed to manage running the model simulations over the spinup period for all your basins, and monitor the jobs in the process. Note that ALL basins must complete their spinup for the program to complete successfully. The spinup program must complete before moving onto the parameter calibration. If you navigate to the 'RUN.SPINUP' subdirectory under your basin, you will see an 'OUTPUT' subdirectory. Under this directory, all files necessary to run the model for your spinup have been constructed here. This includes the namelist files, links to parameter files and executables, along with the spinup output.

Workflow Calibration

You are now ready to calibrate your model parameters for your basins. Fortunately, the process of launching the calibration step is the same as it is with the spinup step. Simply navigate to your calibration python code directory and run the following:

```
python PATH/TO/PyWrfHydroCalib/calibOrchestrator.py JobID --optDbPath
PATH/TO/DATABASE.db
```

Note: `JobID` is the unique job ID value created when you initialized your experiment.

The workflow will then proceed to manage running each calibration iteration, which includes launching model jobs, adjusting parameter files used by the model, running analysis code to calculate new parameter values, and generating updated plots. The workflow will be monitoring each stage of the iteration to ensure things complete successfully. In order for the calibration to complete successfully, the workflow must complete all iterations for every basin in your experiment. The calibration program must complete before you can run the validation program.

Under the 'RUN.CALIB' subdirectory for your basins, you will see a multitude of files and additional

directories compared to the 'RUN.SPINUP' subdirectory. You will see a copy of the calibration parameter table specified during your job initialization, symbolic links to R/Python/executable files, additional subdirectories, a 'calibScript.R' file, and a 'proj_data.Rdata' file. The 'calibScript.R' file is a temporary R namelist file that is updated by the workflow in-between model iterations. This file is used by the R code. The 'proj_data.Rdata' is another temporary file created by R code to hold model statistics and parameter values as they are updated throughout the calibration process. The 'BASELINE_PARAMETERS' directory contains the original parameter files specified during the domain input process. These parameter files include the groundwater parameter file, the 2D Hydro file, along with the Fulldom and soil properties file. The 'DEFAULT_PARAMETERS' directory are these same files, but with the default values applied to them from the calibration parameters table. The 'FINAL_PARAMETERS' directory contains the same adjusted files, but with the final calibrated parameter values applied to them.

Also within the 'RUN.CALIB' directory is a subdirectory called 'plots'. This directory contains several R plots that can be very useful. A more detailed explanation of some of the products in this sub-directory are explained in the 'Understanding Output from the Workflow' section below.

As with the spinup directory, there is an 'OUTPUT' sub-directory containing model output for each iteration. These files are very dynamic and get overwritten during each model iteration that is run.

It is also worth noting that within the database, the 'Calib_Params' and 'Calib_Stats' tables are updated dynamically by the workflow as the calibration progresses. The 'Calib_Params' table will contain the parameter values for each model iteration. The 'Calib_Stats' table contains error metrics for each model iteration as parameter values are adjusted. Many of these values are visualized through the plots that are generated during the calibration process. However, having these values in the database allows you to perform further analysis with the data after the fact to better understand how the system evolved over time.

Workflow Validation

You are now ready to validate your calibrated parameter values for your basins. As you would expect, the process of running the validation is exactly the same as with the spinup and calibration.

```
python                    PATH/TO/PyWrfHydroCalib/runValidOrchestrator.py
PATH/TO/PyWrfHydroCalib JobID --optDbPath PATH/TO/DATABASE.db
```

Note: `JobID` is the unique job ID value created when you initialized your experiment.

The workflow will then proceed to manage running NWM/WRF-Hydro over your basins with the default parameter values specified at the beginning of the experiment, along with your calibrated parameter values. Under the 'RUN.VALID' subdirectory for each basin, you will see several files and sub-directories as you did with the 'RUN.CALIB' directory. There are symbolic links to R scripts and executables used to run analysis. Additionally, there is a 'plots' subdirectory with plots for the user to inspect final analysis for their calibrated parameter values (see section below on Understanding Output from the Workflow). The 'valid_stats.txt' file is a simple text file containing evaluation statistics that are

logged into the ‘Valid_Stats’ table in the database.

Workflow Error and Status Messaging

One of the benefits to running the workflow is the highly simplified nature of initiating the workflow for automation. Upon running any of the main workflow programs (spinup, calibration, validation), the workflow will be managing model output, database entries, and running jobs automatically without user interference. Upon successful completion of the main working programs, the workflow will send the user a message (or print it to screen) indicating the program has finished successfully for all basins.

However, it is possible things could go wrong in the process. The workflow was designed to try to catch all possible problems that may occur, such as:

- The model crashing
- R code that adjusts parameters or performs analysis crashing
- Necessary input files not found

In the event the model crashes, the workflow will see this and send a warning message to you that the model has crashed once for a particular basin. The workflow will then attempt to restart the model based on available restart files. However, if the model crashes again, the workflow will then lock up on the basin and send an error message to you with a path to the created lock file.

NOTE: If a basin in your experiment crashes, you do not need to kill the workflow program you are running, it will continue to run, even if you have a basin that has been locked. However, if you do need to kill the workflow, you can restart it at any time.

In this situation, you will need to determine the cause of the crash in order to proceed. For example, perhaps for this particular basin, there is a missing forcing input file the model is expecting. Upon fixing this issue, simply remove the lock file. The workflow will see the lock file has been removed. It will then restart the model as it did before and continue the workflow process. The same issue could occur with analysis code run to estimate new parameters, evaluate model output, and generate new parameter input files.

Each ‘OUPTUT’ directory for the spinup, calibration, and validation will contain standard output and error log files that are produced by the job submission. If for any reason things fail, it is worth checking those files first to quickly determine what the potential issue is. Also note that during the calibration process, unique status values are logged into the ‘Calib_Stats’ table under the “complete” column. If the workflow is killed during the calibration process, the user should be able to quickly restart without much effort as the workflow pulls these values from the table upon initialization to determine where it needs to work.

Understanding Output from the Workflow

During the various steps of the calibration workflow, there are many statistics and plots generated by the workflow for your benefit. Within the ‘RUN.CALIB’ subdirectory for each basin in your calibration experiment, you will see a ‘plots’ subdirectory. This subdirectory contains several useful plots that are

generated and updated after each calibration iteration. They include visualizations of parameter evolution throughout the calibration, how error metrics evolve, and simulated hydrographs from the different parameter datasets. Sample output is shown below.

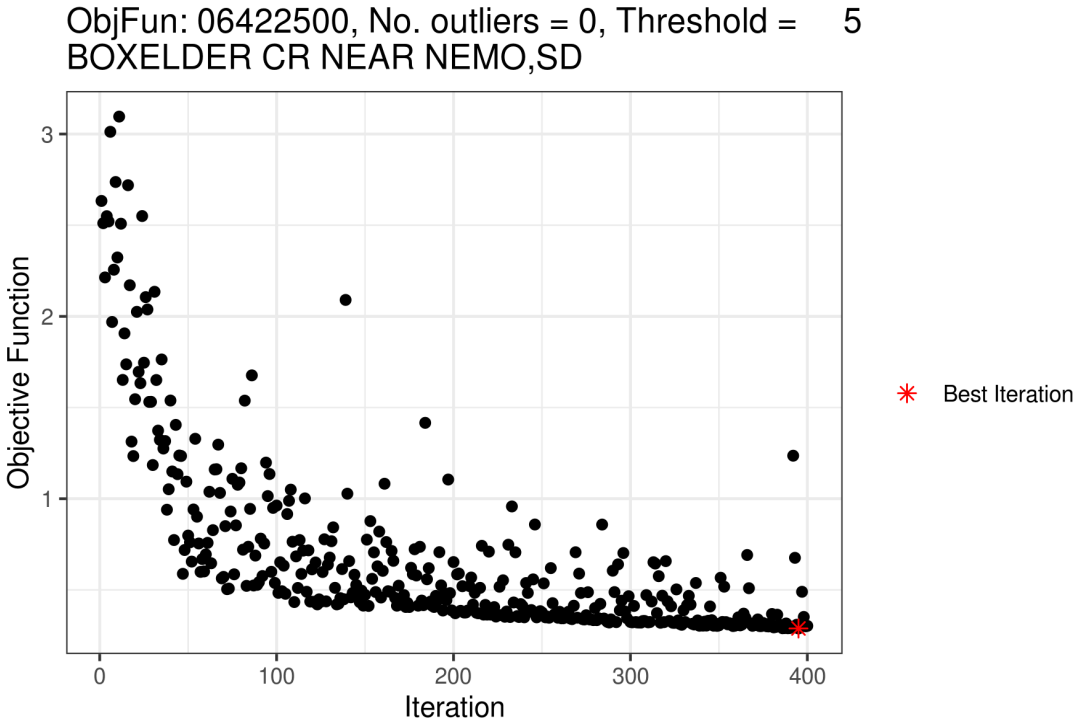

Figure 2. Objective function at different calibration iterations, the best objective function is starred.

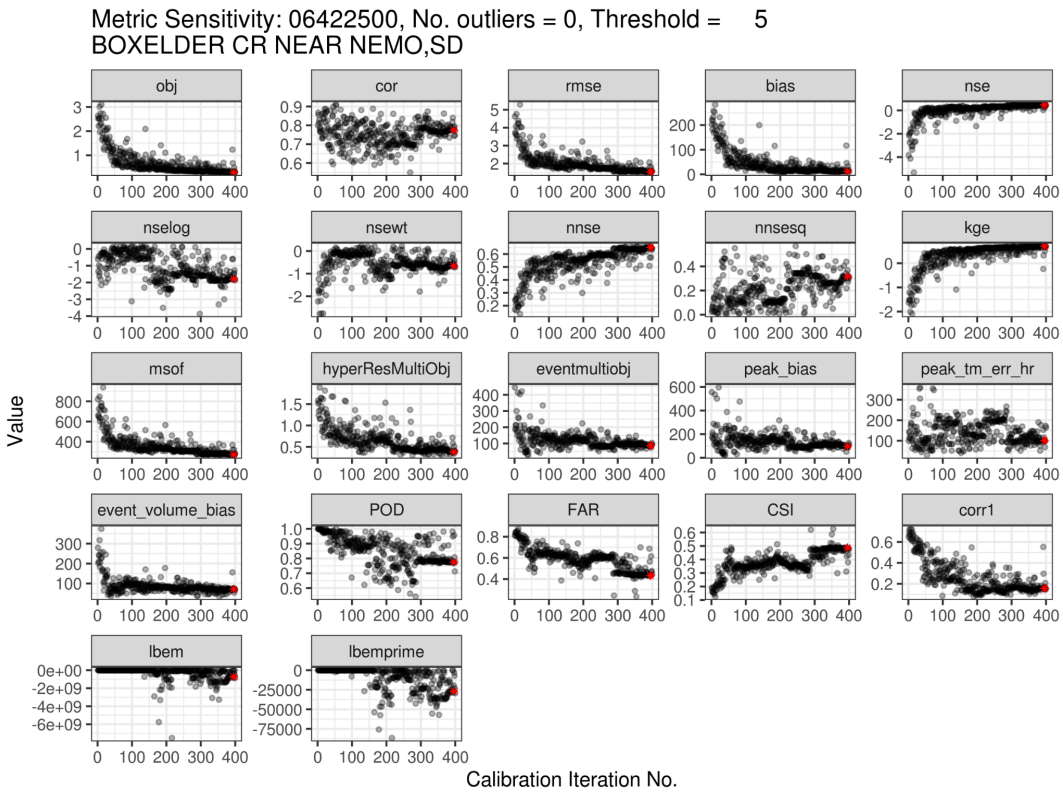

Figure 3. Plot of all the diagnostic metrics as well as the objective function at different calibration iteration. Best iteration is starred.

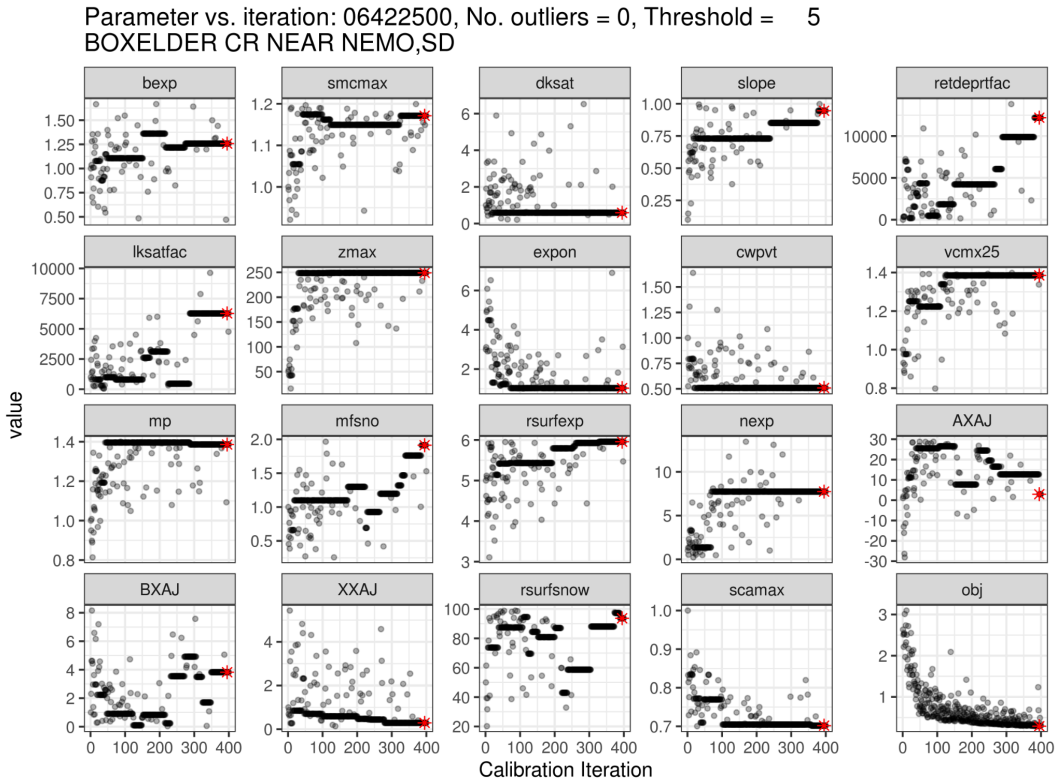

Figure 4. Plot of all the calibrated parameters at different calibration iterations. Best iteration is starred.

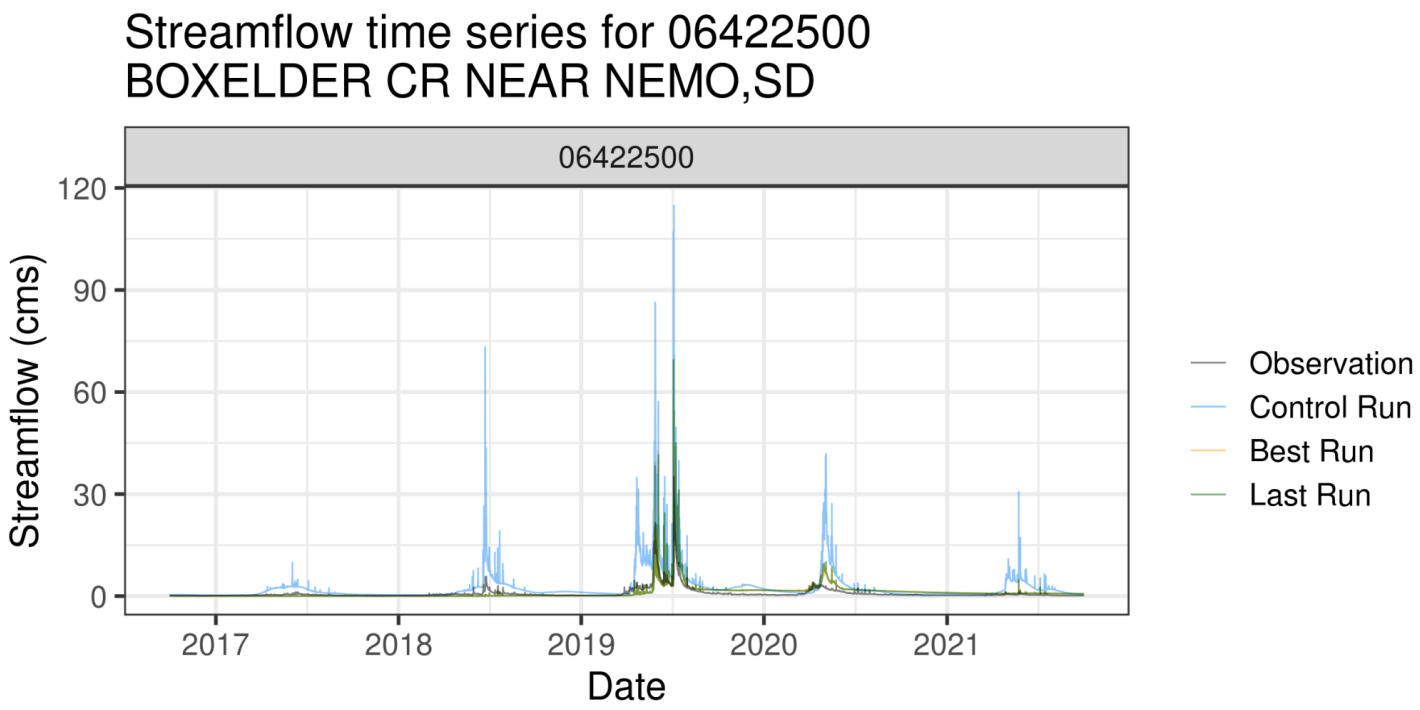

Figure 5. Hydrograph of the best, last and control simulations as well as the observed values.

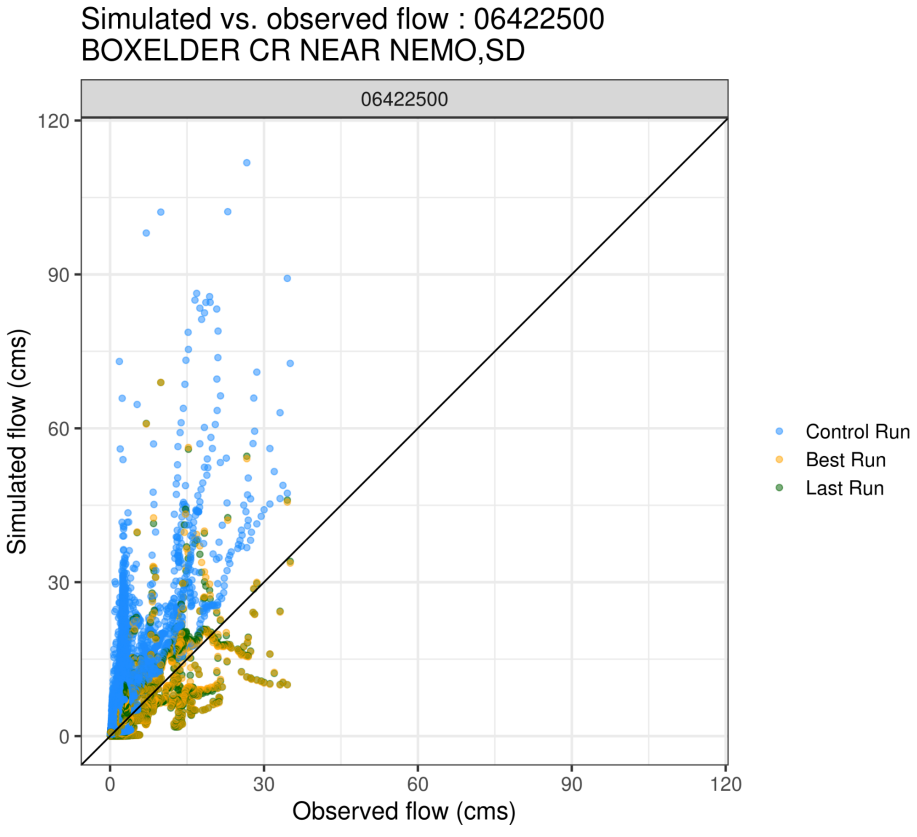

Figure 6. Scatter plot of the best, last and control simulations against observed values.

These plots provide useful insight into parameter/error evolution through the calibration process, which in turn, can provide the user guidance on the behavior of the model in their watershed. In addition to the calibration plots, many of these statistics are entered into the database. Recall the Calib_Params contains the parameter name and value that was determined through the calibration algorithm at the end of each iteration. In addition, the Calib_Stats table contains error statistics for each iteration, along with a flag indicating if that iteration was determined to be the best parameter dataset.

Upon completion of the model validation (after calibration), the workflow generates plots and statistics much in the same way as it did during the calibration step. Under the RUN.VALID subdirectory for a basin, there is a 'plots' subdirectory that contains useful plots visualizing model output for the validation period (see below).

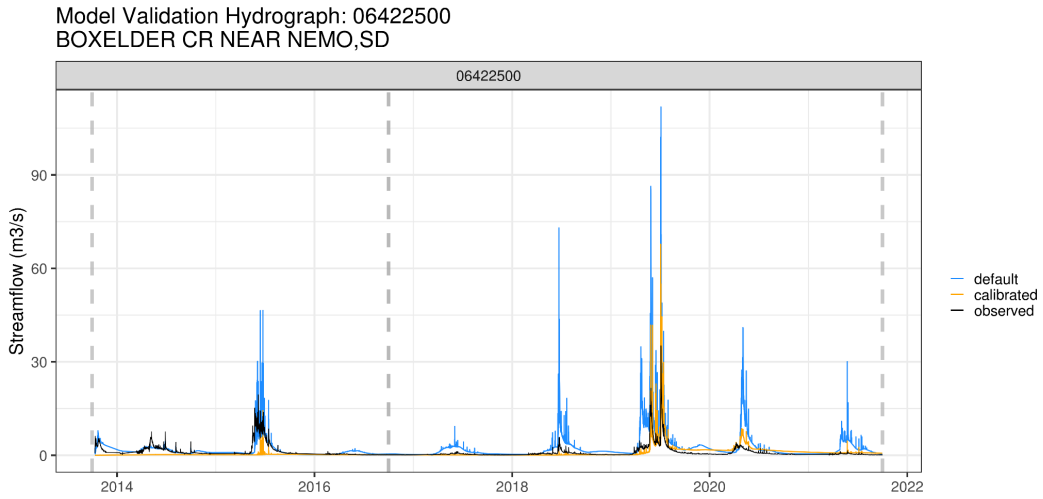

Figure 7. Hydrograph of the best and control simulations as well as the observed values.

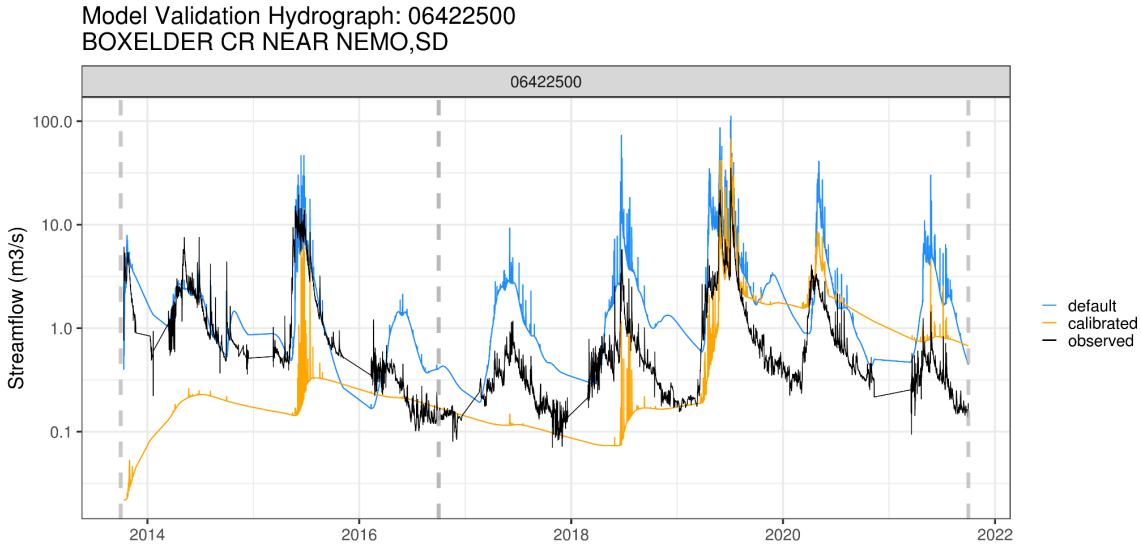

Figure 8. Log Hydrograph of the best and control simulations as well as the observed values.

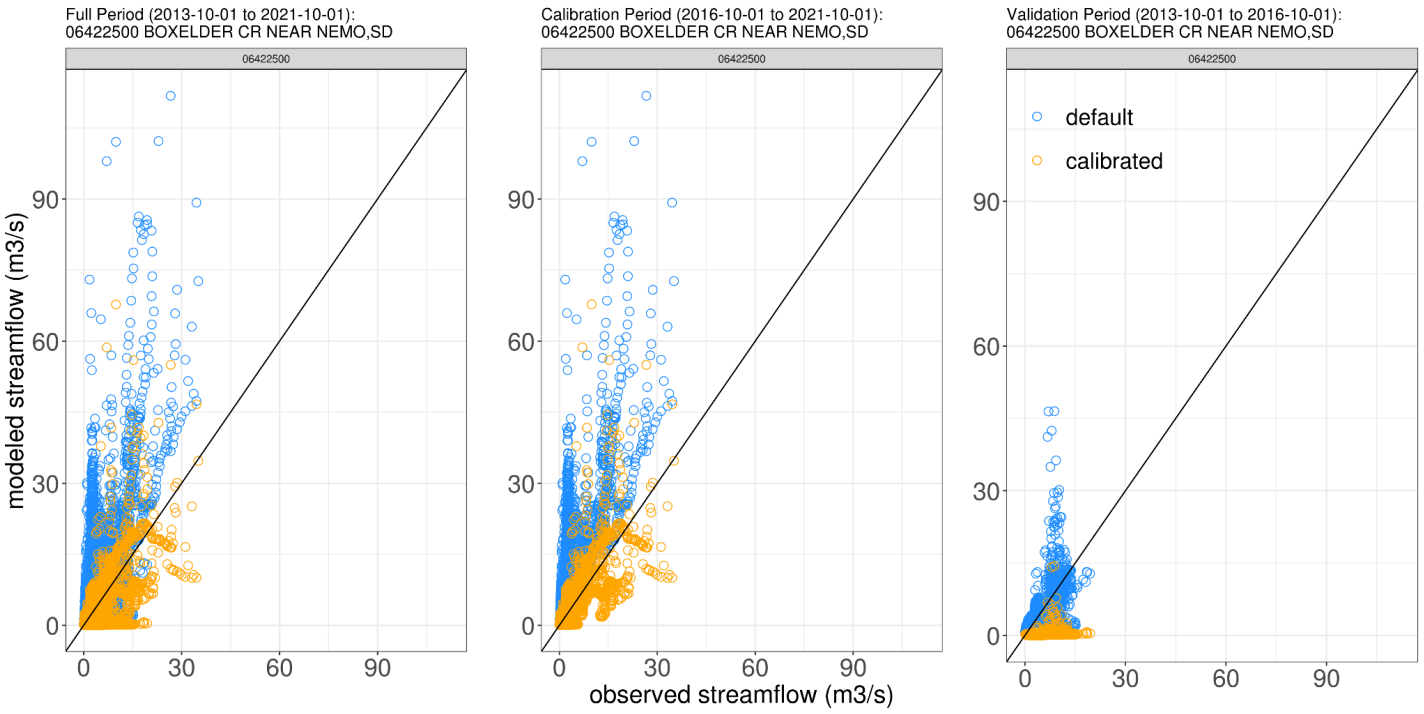

Figure 9. Scatter plot of the best and control simulations as well as the observed values.

Figure 10. Summary of the metric for best and default simulations for calibration (calib), validation (valid) and full duration of calibration and validation (full)

Within the database, you will find these same error statistics for the model simulations in the Valid_Stats table.

This information can be extremely useful when attempting to optimize your watersheds model performance. It can help identify parts of the model that may be more sensitive compared to others. It can also identify which parameters vary more widely compared to others, which in itself can be useful information. It is also possible that you will run a calibration experiment and realize you did not allocate enough iterations for the workflow to properly converge onto a stable set of parameter values. As stated earlier in this document, it is up to you to design and setup their experiment to properly calibrate the model.

Contacts

- wrfhydro@ucar.edu : WRF-Hydro user ticketing
- arezoo@ucar.edu Arezoo RafieeiNasab
- adugger@ucar.edu Aubrey Dugger