

Flexible Model-Agnostic Automatic Calibration Tool for the Next Generation Water Resources Modeling Framework Technical Description and User's Guide

Version 1.0.0

May 3, 2024

Xia Feng

Office of Water Prediction/National Water Center

Table of Contents

1. Overview.....	4
2. Software Installation.....	4
2.1 Install calibration packages.....	4
2.2 Build libraries for ngen and formulations.....	5
2.3 Required data and files.....	5
3. Calibration Execution Steps	5
4. Create Input Data and Files	6
4.1. input.config	6
4.2 ginputfunc.py	7
4.3 noaa_owp.py	8
4.4 create_input.py	8
5. Run Calibration and Validation	8
5.1 calibration.py	8
5.2 validation.py	8
5.3 Scripts in ngen.cal subpackage	8
5.3.1 configuration.py.....	8
5.3.2 strategy.py	8
5.3.3 parameter.py.....	8
5.3.4 agent.py	8
5.3.5 meta.py	9
5.3.6 calibratable.py.....	9
5.3.7 calibration_cathment.py	9
5.3.8 calibration_set.py	9
5.3.9 model.py	9
5.3.10 ngen.py	9
5.3.11 search.py	9
5.3.12 gwo_global_best.py	10
5.3.13 gwo_swarms.py	10
5.3.14 metric_functions.py	10
5.3.15 plot_output.py	10
5.3.16 plot_functions.py	11
5.3.17 validation_run.py	11
5.3.18 utils.py.....	11
5.4 Scripts in ngen.config subpackage	11
5.4.1 cfe.py	11
5.4.2 topmod.py	11
5.4.3 sft.py	11
5.4.4 smp.py	11
5.4.5 lasam.py	12
5.4.6 noahowp.py.....	12
5.4.7 lstm.py	12

5.4.8 sloth	12
5.4.9 pet.py	12
5.4.10 all_formulations.py	12
5.4.11 formulation.py	12
5.4.12 configurations.py	12
5.4.13 bmi_formulation.py	12

1. Overview

This software is developed to automatically calibrate the model parameters within the Next Generation Water Resources Modeling Framework (NextGen). It is composed of three sub-packages written in Python, including *createInput*, *ngen.cal* and *ngen.config*. The *createInput* package generates a variety of input data and configuration files to execute calibration within the NextGen for different model and model component, such as Conceptual Functional Equivalent (CFE), TOPMODEL, Noah-OWP-Modular (NOM), lumped arid and semi-arid (LASAM), soil freeze-thaw (SFT) and soil moisture profiles (SMP). The *ngen.cal* package establishes the job run and performs the parameter search using one of the three optimization algorithms, namely dynamic dimensioned search (DDS), particle swarm optimization (PSO) and grey wolf optimizer (GWO). It also contains various useful utilities, such as statistical evaluation of streamflow, visualization of streamflow, metrics and parameters, and email notification of completeness of job run. Furthermore, it implements functionalities to execute the validation runs using the default and the best calibrated parameters. The *ngen.config* package parses and validates the assorted configurations in the model realization file for different NextGen formulations. Figure 1 shows a general picture of workflow to create input data and files, and execute the calibration and validation runs using this calibration capability.

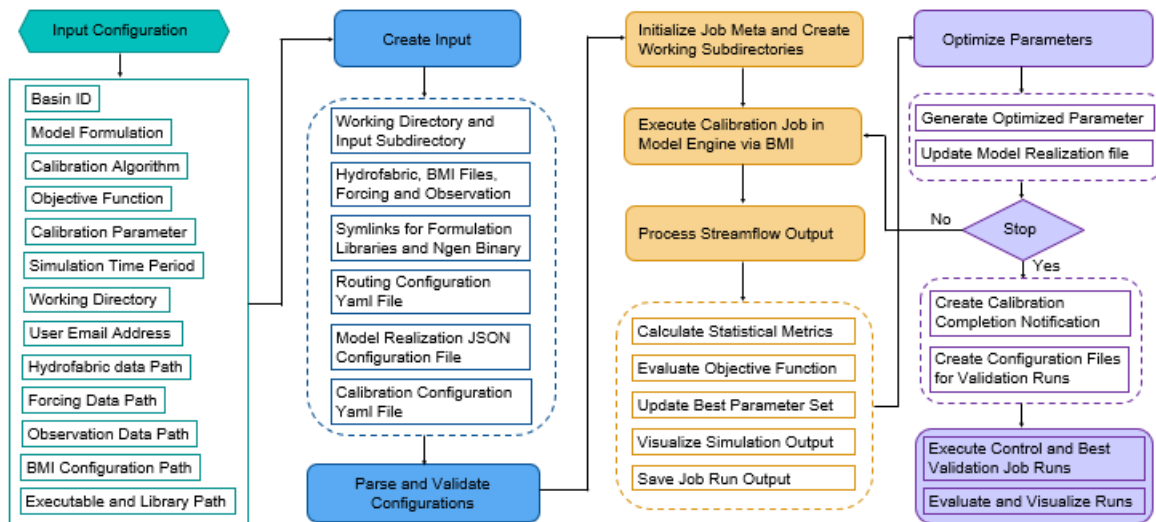


Figure 1. Schematic of the calibration and validation workflow.

2. Software Installation

2.1 Install calibration packages

The source codes of three Python subpackages are available from the GitHub repository at https://github.com/NOAA-OWP/NextGen_Model_Calibration. They can be downloaded to the local Linux terminal using the following command

```
git clone https://github.com/NOAA-OWP/NextGen_Model_Calibration
```

Create and activate Python virtual environment to work on your project

```
python -m venv venv  
source venv/bin/activate
```

Then go to `createInput` directory and install all three subpackages alongside the other dependencies into the virtual environment `venv`

```
cd NextGen_Model_Calibration/python/createInput  
pip install .
```

2.2 Build libraries for `ngen` and formulations

The calibration workflow is executed using the `ngen` engine through the NextGen Basic Model Interface (BMI). The detailed instructions to download and build libraries for `ngen` and formulations can be found at <https://github.com/NOAA-OWP/ngen/wiki>.

2.3 Required data and files

To conduct calibration through `ngen` engine, you will need to select the river basins, NextGen formulations, model parameters, and prepare the data and files as follows

- *Hydrofabric files*: Geopackage files subset for the river basins or catchments
- *Meteorological forcing data*: They are subset from AORC data or other sources of data for the chosen basins and time period to drive the calibration simulations
- *Streamflow observaton data*: They can be downloaded prior to performing calibration, otherwise the calibration workflow has the option to download the observed streamflow from NWIS data portal on the fly.
- *Calibration parameters file*: Initial, minimum and maximum values of formulations

3. Calibration Execution Steps

There is a master configuration file `input.config` located in directory `createInput/examples/config`

The first step is to fill out this template file with the required fields related to the calibration setup, data, files and libraries, etc. The entry for each field is explained in the file with the comments and will be described in the next section. Then create the input data and files for the selected basin, e.g., `01010101`

```
python create_input.py input.config
```

Under the working directory `workdir`, basin directory `01010101` is created. It contains model realization file and subdirectory `Input` wherein all input data and configuration files including calibration configuration file, `01010101_config_calib.yaml` reside.

Then execute calibration job run

```
python calibration.py  
workdir/01010101/Input/01010101_config_calib.yaml
```

Directory `Output` is created under the basin directory `01010101` with subdirectories, `Calibration_Run` and `Validation_Run` to save all the output files for the calibration and validation run, respectively.

When calibration job is completed, an empty log file is created in `Calibration_Run`. Moreover, the realization files and configuration files for the control and best validation run will be created in directory `Validation_Run`. The control and best validation jobs can be launched with the following two commands:

```
python validation.py workdir/01010101/Output/Validation_Run  
/01010101_config_valid_control.yaml
```

```
python validation.py workdir/01010101/Output/Validation_Run  
/01010101_config_valid_best.yaml
```

Similarly, log flag files will be generated after control and best validation runs are completed. And you will receive email notification once calibration or validation run is finished if your email address is provided in `input_config` file.

4. Create Input Data and Files

Subpackage *createInput* subsets and creates a variety of data and configuration files needed to setup and run the calibration workflow.

4.1. *input.config*

This is the main configuration file comprising many options in three sections, including General, Calibration and DataFile as shown in Table 1.

Table 1. Description of Options in the Configuration File.

Name	Description	Comments
Basin	Basin ID	
model	Model formulation	
run_type	Calibration or validation	
main_dir	Main working directory	
optimization_algorithm	Selected optimization algorithm	
swarm_size	Population number	For PSO or GWO
c1	Cognitive coefficient	For PSO
c2	Social coefficient	For PSO
w	Inertial weight	For PSO
objective function	Objective function	

start_iteration	Starting iteration number	
number_iteration	Total number of iterations	
restart	Whether to restart calibration run	
calib_start_period	Starting time for calibration run	
calib_end_period	Ending time for calibration run	
calib_eval_start_period	Evaluation starting time for calibration run	
calib_eval_end_period	Evaluation ending time for calibration run	
valid_eval_start_period	Evaluation starting time for validation run	Optional
valid_eval_end_period	Evaluation ending time for validation run	Optional
full_eval_start_period	Evaluation starting time for full time period run	Optional
full_eval_end_period	Evaluation ending time for full time period run	Optional
save_plot_iteration	Whether to save figures at each iteration	Optional
save_output_iter	Whether to save streamflow at each iteration	Optional
Save_plot_iter_freq	Iteration interval to save plots	Optional
streamflow_threshold	Thresholds for calculation of categorical scores	Optional
station_name	Gage station name	Optional
user_email	User email address	Optional
forcing_dir	Directory for meteorological forcing data	
obs_dir	Directory for observation data	
hydrofab_dir	Directory for hydrofabric files	
cfe_dir	Directory for CFE initial BMI configuration files	Optional
topmod_dir	Directory for TOPMODEL initial BMI configuration files	
noah_parameter_dir	Directory for NOM parameter tables	Optional
Attributes_file	Path to model parameter attributes file	
calib_parameter_file	Path to calibration parameter file	Optional
lasam_soil_parameter_file	Path to LASAM soil parameter table	Optional
lasam_soil_class_file	Path to LASAM soil class file	Optional
ngen_exe_file	Path to ngen executable file	Optional
cfe_lib	Path to CFE library file	Optional
sloth_lib	Path to SLoTH library file	Optional
topmod_lib	Path to TOPMODEL library file	Optional
noah_lib	Path to NOM library file	Optional
sft_lib	Path to SFT library file	Optional
smp_lib	Path to SMP library file	Optional
lasam_lib	Path to LASAM library file	Optional

4.2 ginputfunc.py

This module contains the following functions:

- *create_walk_file*: Generate crosswalk file for the specified basin
- *create_cfe_input*: Create initial BMI configuration files for CFE
- *create_noah_input*: Create initial BMI configuration files for NOM
- *create_sft_smp_input*: Create initial BMI configuration files for SFT and SMP
- *create_lasam_input*: Create initial BMI configuration files for LASAM
- *change_topmodel_input*: Change path of initial BMI configuration files for TOPMODEL

- *create_routing_config*: Create configuration file for t-route model
- *create_realization_file*: Create ngen model realization file with different formulations
- *create_calib_config_file*: Create calibration configuration file

4.3 noaa_owp.py

This module generates BMI initial configuration files for NOM.

4.4 create_input.py

This is the main code to read `input_config`, extract hydrofabric, forcing data and streamflow observation for the selected basin and generate BMI initial configuration files, t-route configuration file, model realization file and calibration configuration file for the chosen formulations to execute the calibration and validation runs.

5. Run Calibration and Validation

The subdirectory `runCalibValid` contains two execution scripts `calibration.py` and `validation.py` and two subpackages *ngen.cal* in `ngen_cal/src/ngen/cal` and *ngen.config* in `ngen_config/src/ngen/config`.

5.1 calibration.py

This is the main program to read calibration configuration yaml file, initialize calibration job run and execute calibration simulation with the specified optimization algorithm and calibration strategy.

5.2 validation.py

Main program to read validation configuration yaml file, initialize validation job run and execute validation simulations using the default and best parameter set, respectively.

5.3 Scripts in ngen.cal subpackage

5.3.1 configuration.py

It defines different fields for the *general* configurations of the configuration yaml file.

5.3.2 strategy.py

This module defines optimization algorithms, calibration objective functions, calibration strategies and sensitivity strategy.

5.3.3 parameter.py

This module defines different calibration parameters with name, initial, minimum and maximum values.

5.3.4 agent.py

This module defines and creates working directory and subdirectories for the calibration and validation job runs plus defining configuration and calibration related variables.

5.3.5 *meta.py*

It creates job directory for different runs and defines the job log file.

5.3.6 *calibratable.py*

This script defines various output files such as streamflow at best and last iteration, metrics, objective function, parameters, etc alongside the other configuration fields used for calibration and evaluation. It also contains functions to process calibration parameters and save different output files.

5.3.7 *calibration_cathment.py*

This module creates interface to adjust parameters, store output and observation and perform evaluation for catchments.

5.3.8 *calibration_set.py*

Similar to above script but this module is used for a set of the catchments. Additionally, it contains functions to save streamflow output from calibration at the specified and best iteration. It also saves the streamflow output from the validation runs.

5.3.9 *model.py*

It defines and stores configurations, output files and other variables used in the calibration and evaluation processes. It defines the functions to update the calibration states and write different output files, like metrics, cost function, calibrated parameters, plus parameter sets from swarm based global optimization algorithms. Moreover, it defines functions to restart the terminated calibration run, such as saving copies of the files for objective function, metrics and parameters, cleanup the original files and restart calibration from the previous iteration. It also includes functions to create realization and configuration files for the validation control and best runs using the default and the best calibrated parameter set, respectively.

5.3.10 *ngen.py*

This script contains functions to read and parse all kinds of configurations from the input configuration yaml and model realization json file. It reads and stores the hydrofabric file and meteorological forcing file. It also includes functions to process calibration parameters, update realization file together with other configurations used in the model runs. Moreover, the script extracts the hydrolocation for different calibration strategies.

5.3.11 *search.py*

This script executes model run, evaluates streamflow output, searches the best parameter set, updates realization file and saves different intermediate output variables until all the iterations are completed. It also creates configuration and model realization files for two validation runs. Moreover, it writes an empty log file to signify the completion of calibration or validation runs and sends email notification to the user if the email address is provided in the input configuration file.

5.3.12 *gwo_global_best.py*

It includes the main function to optimize the model parameters using GWO algorithm.

5.3.13 *gwo_swarms.py*

This script contains supplementary functions used to perform GWO in the above module. Additionally, it includes functions to write and read the intermediate cost function and parameter sets during the calibration. These files can be used for restarting calibration run if the iteration stops for some reasons.

5.3.14 *metric_functions.py*

This script evaluates simulated streamflow output against the observed streamflow and calculates sixteen statistical metrics as shown in Table 2.

Table 2. Description of Statistical Metrics.

Metric	Description	Function
Cor	Pearson correlation	pearson_corr
MAE	Mean absolute error	mean_abs_error
RMSE	root mean square error	root_mean_squared_error
RSR	Ratio of RMSE to standard deviation of observation	rmse_std_ratio
PBIAS	Percentage bias	percent_bias
KGE	Kling-Gupta-Efficiency	KGE
NSE	Nash-Sutcliffe-Efficiency	NSE
LogNSE	Logarithmic form of NSE	NSE
NSEWt	Weighted average of NSE and LogNSE	Weighted_NSE
POD	Probability of detection	categorical_score
CSI	Critical success index	categorical_score
FAR	False alarm ratio	categorical_score
FBIAS	Frequency of bias	categorical_score
HFDC	PBIAS of high flow of flow duration curve (FDC)	pbias_fdc
MFDC	PBIAS of slope of FDC	pbias_fdc
LFDC	PBIAS of low flow segment of FDC	pbias_fdc

5.3.15 *plot_output.py*

This is the script to read different output files from the calibration or validation runs and create eight types of figures during calibration and four kinds of figures during validation.

5.3.16 *plot_functions.py*

This visualization script includes the following functions:

- *plot_streamflow*: Time series of streamflow from control run using the default parameter set, best run using the best calibrated parameter set and last run from the calibrated parameter set at the current iteration compared with the observation
- *plot_streamflow_precipitation*: Similar to above but overlapped with precipitation
- *plot_fdc_calib*: FDC from control, best and last run compared with the observation
- *plot_fdc_valid*: Similar to above but for validation control and best run
- *scatterplot_streamflow*: Scatterplot of the simulated streamflow from control, best and last run relative to the observed streamflow
- *scatterplot_objfun*: Scatterplot of objective function at each iteration
- *scatterplot_var*: Scatterplot of statistical metrics or calibrated parameters at each iteration
- *scatterplot_objfun_metric*: Scatterplot of statistical metrics relative to objective function
- *barplot_metric*: Barplot of statistical metrics from validation control and best run
- *plot_cost_hist*: Cost function from global and local best position at each iteration from PSO or GWO algorithm

5.3.17 *validation_run.py*

This is the main script to execute validation control and best run using the default and best calibrated parameter set, respectively. It calculates and saves the statistical metrics, writes the streamflow output, plots the various files as well as generate the log file and sends email notification if the validation runs are completed.

5.3.18 *utils.py*

It includes utilities like changing the working directory and sending email notification to the user if the run is completed.

5.4 Scripts in *ngen.config subpackage*

5.4.1 *cfe.py*

It defines the BMI configurations and calibration parameters for cfe.

5.4.2 *topmod.py*

Similar to above but for TOPMODEL.

5.4.3 *sft.py*

Similar to above but for SFT model.

5.4.4 *smp.py*

Similar to above but for soil moisture profiles.

5.4.5 *lasam.py*

Similar to above but for lumped arid and semi-arid model.

5.4.6 *noahowp.py*

Similar to above but for Noah-OM.

5.4.7 *lstm.py*

It define the configurations for long short-term memory model.

5.4.8 *sloth*

Similar to above but for SLoTH.

5.4.9 *pet.py*

Similar to above but for potential evapotranspiration formulations.

5.4.10 *all_formulations.py*

It collects all above formulation modules.

5.4.11 *formulation.py*

It defines several configurations for different formulations.

5.4.12 *configurations.py*

It defines forcing, time and routing configurations in realization file.

5.4.13 *bmi_formulation.py*

This script defines the configurations used for executing different ngen formulations through BMI and functions to validate the configurations.