Appendices of

The benefits and trade-offs of multi-variable calibration of WGHM in the Ganges and Brahmaputra basins

H. M. Mehedi Hasan et al.

Correspondence to: H.M. Mehedi Hasan (mehedi.hasan@gfz-potsdam.de)

Appendix A: Elementary Effect Test (EET) method of Morris (1991)

A1.1 Background

Morris's method of sensitivity analysis, also known as the Elementary Effect Test (EET) method, involves calculating local derivatives of a response variable y (where $y=f(\theta)$; $\theta \in R^m$; m being the number of input parameters) with respect to each parameter θ_i, referred to as the Elementary Effect (EE) of θ_i, at multiple random reference points $\theta_{\mathrm{ref},j}$ (where $j \in \{1,2,\ldots,r\}$; r being the total number of elementary effects for a single parameter) in the parameter space Ω (where $\Omega \subseteq R^m$). The elementary effect of the i^{th} parameter is computed as follows:

$$EE_{(i,j)} = \frac{f(\theta_{(per,i,j)}) - f(\theta_{(ref,j)})}{\Delta_{(i,j)}} \quad \text{(I)}$$

where, $\theta_{(ref,j)} = (\theta_{(1,j)}, \theta_{(2,j)}, \ldots, \theta_{(i,j)}, \ldots, \theta_{(m,j)})$ is the j^{th} reference parameter set and $\theta_{(per,i,j)} = (\theta_{(1,j)}, \theta_{(2,j)}, \ldots, \theta_{(i,j)} + \Delta_{(i,j)}, \ldots, \theta_{(m,j)})$ is the perturbed parameter set where the i^{th} parameter has been perturbed by $\Delta_{(i,j)}$.

The average of the elementary effects is expressed as the sensitivity index (SI) of the i^{th} parameter as follows,

$$SI_i = C_i \times \frac{1}{r}\sum_{j=1}^{r} EE_{(i,j)} \quad \text{(II)}$$

C_i is a scaling constant used for intercomparison of the indices (*SI*) among parameters and enables ranking of these indices. Typically, the parameter range ($RANGE_i = MAX_i - MIN_i$) is used as the scaling constant. The method of Morris also produces an unbiased estimator of the variance S^2 that reflects the combined effect of nonlinearity of the model response and interactions among input parameters (Morris, 1991).

$$S_i^2 = VAR(EE_{(i,j)}) \quad \text{(III)}$$

The unbiased standard error of the mean, i.e., the standard error of the sensitivity index, can be estimated as follows.

$$SEM_i = \frac{S_i}{\sqrt{r}} \quad \text{(IV)}$$

Campolongo et al. (2011) proposed a slight modification of (II) to use absolute elementary effect, which is necessary for non-monotonic models, e.g.,

$$SI_i^* = C_i \times \frac{1}{r}\sum_{j=1}^{r}\left|EE_{(i,j)}\right| \quad \text{(V)}$$

In the current study, we followed the sampling design proposed by Campolongo et al. (2011). However, the modification described in (V) was not applied in this study for two reasons: (1) our model response was not a scalar value but rather a time-series, and (2) the measure of change in the target response variable was computed as the Root Mean Squared Deviation (RMSD) of model responses between perturbed and reference parameter sets, only after both runs had finished. Because the model response consists of a monthly time-series, the RMSD was computed as follows:

$$RMSD = f(\theta_{per}) - f(\theta_{ref}) = \sqrt{\frac{1}{N}\sum_{k=1}^{N}\left(S_{(per,k)} - S_{(ref,k)}\right)^2} \quad \text{(VI)}$$

where $S_{(per,\,k)}$ and $S_{(ref,\,k)}$ refer to simulated model responses at the k^{th} point in time (i.e., month) with the perturbed parameter set (θ_{per}) and the reference parameter set (θ_{ref}) and N is the number of time points in the response time-series.

The total number of model runs required by the EET methods is $r \times (m + 1)$.

A1.2 Workflow

Below, we present the workflow of the EET sensitivity analysis, adapted from the workflow presented in Pianosi et al. (2015). In the following algorithms, vectors and matrices are shown in boldface, indices are presented in parentheses, and procedures are described with parentheses in regular font.

Algorithm 1: Workflow of EET Sensitivity Analysis

Step 1: Create the EET design matrix $\boldsymbol{X}$ following the radial-design of Campolongo et al. (2011), as in **Algorithm 2**. The size of $\boldsymbol{X}$ is n × m; n := $r \times (m + 1)$ is the number of sample, m is the number of parameters, and r is the number of EEs to be computed.

Step 2: Run the simulation model with each parameter set represented by each row of $\boldsymbol{X}$ and compute $\boldsymbol{Y} = RMSD(S_1, S_2)$ as in Equation (VI)

Input: $\boldsymbol{X}$:= the sampling matrix
Output: $\boldsymbol{Y} := f(\boldsymbol{X})$, the model response which is the *RMSD* between two runs
Set $n \leftarrow$ no. of rows in $\boldsymbol{X}$
Create a matrix $\boldsymbol{Y}$ with size $n \times 1$
Set $i \leftarrow 0$
For $i \leftarrow 0$ to $(n - 1)$ do
    $\theta \leftarrow X(i, :)$
    $S_1 \leftarrow WGHM(\theta)$, *given WGHM is a predefined procedure that describes the WaterGAP GHM*
    If i is divisible by $(m + 1)$ then

$S_2 \leftarrow S_1$
$Y(i) \leftarrow 0$
Else do
$Y(i) \leftarrow RMSD(S_1, S_2)$, *given RMSD procedure computes the root mean squared deviation between two simulated time-series*
End if
End for

Alternative approach (for scalar model response):
Input: $\boldsymbol{X}$:= the sampling matrix
Output: $\boldsymbol{Y} := f(\boldsymbol{X})$, the scalar model response

Set $n \leftarrow$ no. of rows in $\boldsymbol{X}$
Create a matrix $\boldsymbol{Y}$ with size $n \times 1$
Set $i \leftarrow 0$
For $i \leftarrow 0$ to $(n\text{-}1)$ do
$\theta \leftarrow X(i, :)$
$Y(i) \leftarrow f(\theta)$, given f(.) is the procedure defining the model
End for

Step 3: Compute EET Indices

Input: $\boldsymbol{X}$:= the sampling matrix, $\boldsymbol{Y} := f(\boldsymbol{X})$, ***RNG*** := a vector of size m with parameter ranges
Output: ***mi*** := a vector of the mean sensitivity indices of m parameters, **sigma** := a vector of standard deviations of sensitivity indices for all parameters

Set $r \leftarrow$ no. of elementary effects, m $\leftarrow$ no. of parameters, n $\leftarrow$ r * (m + 1)
Set ***mi*** $\leftarrow$ a vector of m zeros, **sigma** $\leftarrow$ a vector of m zeros, **EE** $\leftarrow$ an r-by-m matrix of zeros

Set $i \leftarrow 0$, $k \leftarrow 0$
While $i < n$ do

Set $j \leftarrow 0$
For $j \leftarrow 0$ to $(m\text{-}1)$ do
$\Delta\theta \leftarrow \boldsymbol{X}(i, j) - \boldsymbol{X}((i + j + 1), j)$
$\Delta Y \leftarrow \boldsymbol{Y}(i) - \boldsymbol{Y}(i + j + 1)$

EE $(k, j) \leftarrow |\Delta Y / \Delta\theta| * RNG(j)$
End for

$k \leftarrow (k + 1)$
$i \leftarrow i + (m + 1)$
End while

For $j \leftarrow 0$ to $(m\text{-}1)$ do
mi (j) $\leftarrow$ *mean*(EE(:,j)), given the procedure *mean* (...) computes the mean

sigma(j) ← *std*(EE(:,j)), given the procedure *std(...)* computes standard deviation

End for

Step 4: Select influential parameters according to **Algorithm 3**

Algorithm 2: Generation of EET Sampling Matrix following the radial design of Campolongo et al. (2011)

Input: r := number of elementary effects per parameter, m := number of parameters, **etc** := parameter PDFs and additional information

Output: **X** := an n-by-m sampling matrix where $n = r * (m + 1)$

Step 1: Generate reference and auxiliary samples using Latin Hypercube Sampling (LHS) strategies. The procedure *lhcube(...)* produces an n-by-m sample matrix, where n is the number of samples and m is the number of parameters, respecting probability distribution functions (PDF) of each parameter. The lhcube(…) procedure requires three arguments: (i) n, (ii) m, and (iii) ***etc*** := parameter PDFs and additional information. See the SAFE toolbox of Pianosi et al. (2015) for detailed implementation of *lhcube(...)* procedure and its usage for drawing samples from parameter PDFs.

Set n ← r * 2
AB ← *lhcube*(n, m, **etc**)
Set ***A*** ← the first half of ***AB*** as the reference or baseline points
Set ***B*** ← the last half of ***AB*** as the auxiliary points. The auxiliary points is used to deviate parameter values from baseline one at a time

Step 2: Create the sampling matrix ***X*** using the r-by-m baseline matrix ***A*** and r-by-m auxiliary matrix ***B***. Each parameter value in a baseline sample is replaced one at a time by the respective parameter value in an auxiliary sample to produce m perturbed samples for each reference sample. Finally, stack all perturbed samples along with the reference of the baseline sample to form the final sampling matrix ***X***.

Set n ← r * (m + 1), ***X*** ← an n-by-m matrix with zero values
Set i ← 0
For k ← 0 to (r – 1) do
    Set ***a*** ← ***A***(k, :), ***b*** ← ***B***(k, :)
    Set ***c*** ← m-by-m matrix by replicating the ***a*** vector m times
    For j ← 0 to (m – 1) do **c** (j, j) ← **b**(j) End
    X(i, :) ← **a** i.e., copy all elements of ***a*** into the i^{th} row of **X**
    For j ← 0 to (m – 1) do ***X***((i+j+1), :) ← ***c***(j, :) End
    i ← i + (m + 1)
End for

A1.3 Parameter Selection

In the multi-variable, multi-signature setting of sensitivity analysis, our goal was to select parameters that exerted sufficient influence on any of the target response variables across all chosen signatures. Therefore, we individually selected influential parameters for each variable and for each signature, and then we aggregated these selections across all cases. Additionally, during the selection process, we considered the uncertainty associated with sensitivity indices and implemented a Monte Carlo simulation procedure to repeatedly run the selection process, thereby producing a robust solution for parameter selection.

For practical reasons and to limit the number of selected parameters, we employed a selection strategy that ensured a minimum share of all effects produced by all parameters for a response variable and its associated signature were accounted for by the selected parameters. In this study, we chose a threshold of at least 50% of the combined total effect to define the minimum share. The selection algorithm is presented in Algorithms 3 and 4.

Algorithm 3: Parameter selection for Multi-variable Multi-signature SA

Input: mi := mean indices for each variable and for each signature, sigma := the standard deviation of the sensitivity indices, r := no. of elementary effects considered to compute the mean index, th := selection threshold, n := number of Monte Carlo runs

Output: S := a vector of true/false indicating which parameters should be selected

Set S ← a vector of Booleans initialized to false
For each response variable do
    For each signature do
        mi ← get mean sensitivity indices for the target variable and signature
        sigma ← get standard deviation of the indices

        Set ***sem*** ← a vector of *m* zeros
        For j ← 0 to (m – 1) do ***sem***(*k*) ← ***sigma***(*k*) / sqrt(*r*) End

        s ← *ParameterSelection*(***mi***, ***sem***, *th*, *n*), the procedure *ParameterSelection*(…) finds top influential parameter. The procedure details are provided in **Algorithm 4**.

        For *j* ← 0 to (*m* – 1) do
            If ***s***(*j*) = true then ***S***(*j*) ← true
        End for
    End for
End for

Algorithm 4: ***S*** = *ParameterSelection* (***mi***, ***sem***, *th*, *n*); a procedure to find top influential parameters that cover at least a given share of the combined total effect, accounting for the uncertainty in the sensitivity indices

Input: ***mi*** := mean sensitivity indices, ***sem*** := standard errors of the mean indices, th := selection threshold of cumulative effect, n := no. of Monte-Carlo runs

Output: ***S*** := a vector of *m* true/false values representing whether each parameter should be selected

Step 1: Generate *n* set of sensitivity indices by Monte Carlo simulation

Set m ← size of ***mi***
Set **MM** ← an *n*-by-*m* matrix of zeros
For j ← 0 to (m – 1) do
low ← ***mi***(*j*) – ***sem*** (*j*)
high ← **mi**(*j*) + ***sem*** (*j*)
MM(:, *j*) = *uniform*(*low*, *high*, *n*) i.e., fill each column of ***MM*** with random *n* values uniformly distributed between the 'low' and the 'high'. The procedure *uniform(...)* produces random values from a uniform distribution.
End for

Step 2: For each set of sensitivity indices, sort the parameters by their effects on a response variable and compute share of each parameters effect to the sum of all effects. Compute cumulative effect of the sorted parameter list and finally select parameters sequentially until the cumulative effect reaches or exceeds the specific threshold

Set ***S*** ← a Boolean vector of size *m*, initialized to *false*
Set ***a*** ← a vector of *m* zeros, ***e1*** ← a vector of *m* zeros, ***e2*** ← a vector of *m* zeros
For i ← 0 to (n – 1) do
a ← MM(i, :)
e1 ← ***a****(:)*/*sum*(***a***) i.e., the relative effect for each parameter
jj ← indices of sorted ***e1*** in descending order
e2(jj) ← *cumsum*(***e1***(***jj***)) i.e., given the cumsum(…) procedure computes the cumulative sum of a vector, produce the cumulative sums with the sorted ***e1*** values and store them in proper order in ***e2***.

Set ***temp*** ← ***e2***(***jj***), *k* ← 0
While *k* < m do
i_sel ← ***jj***(*k*)
S(*i_sel*) ← true i.e., select high influential parameters until cumulative effect reaches or exceeds the threshold
If ***temp***(*k*) >= *th* then
break
End if
k ← *k* + 1
End while
End for