Swarm Intelligence Approach for Parameter Tuning of a Nonlinear River Flood Routing Model

Rebeca Sánchez[1], Patricia Suárez[1], Akemi Gálvez[1,2], Andrés Iglesias[1,2,†]

[1]Department of Applied Mathematics and Computational Sciences,
University of Cantabria, Avenida de los Castros s/n, 39005, Santander, Spain
[2]Department of Information Science, Faculty of Sciences, Narashino Campus,
Toho University, 2-2-1 Miyama, 274-8510, Funabashi, Japan
[†]Corresponding Author: `iglesias@unican.es`
`http://personales.unican.es/iglesias`

Abstract. The Muskingum model is a very popular flood routing method to estimate the behavior of a river in the event of strong rains or storms that can potentially lead to flooding or landslides. As such, it is widely used for flood prediction and many other tasks. Applying this method to real-world scenarios is not an easy task, as it requires a proper parameter tuning of this model, which is also assumed to be nonlinear for higher accuracy. Moreover, the optimal parameter values of the Muskingum model depend on the physical characteristics of the river reach, and hence, their determination is problem dependent. In this paper we apply a powerful swarm intelligence approach called bat algorithm to solve this parameter tuning problem for nonlinear river flood routing. The method is applied to a known example in the field with good results.

Keywords: Swarm intelligence · bat algorithm · river flood routing · nonlinear models · parameter tuning

1 Introduction

Flood routing is a mathematical technique used to predict the changes over the time in magnitude, speed and shape of a flood wave when water moves in a river or a reservoir. This problem becomes very important in water engineering for predicting landslides and urban flood, among other important issues. Flood routing methods can be roughly classified into two groups: hydraulic methods and hydrologic methods. Hydraulic methods are generally more accurate but hydrologic methods are also widely used due to their higher simplicity and the fact that their solutions are still acceptable in many practical situations. Among the hydrologic methods, the Muskingum model remains one of the most popular thanks to its simplicity. The Muskingum model was proposed by McCarthy for the river of that name in Ohio [8].

Suppose that we have a river reach, that is, a continuous section of a river or stream between an upstream and downstream location having similar hydrologic conditions (e.g., discharge, slope, area) along the way, meaning that the flow measured at a point somewhere along the section is representative of conditions in that section of river or stream. In other words, the whole section can be hydrologically described by the same mathematical model and for the same parameters all along the way. Given the inflow at the upstream end of the river reach, the outflow at the downstream end is given by:

$$\frac{I_1 + I_2}{2} - \frac{O_1 + O_2}{2} = \frac{S_1 + S_2}{\Delta t}$$

where I is the inflow rate to the reach, O is the outflow, S is the storage (the volume of water stored), Δt is the time increment, and subscripts 1 and 2 denote the values of those variables at the beginning and at the end of the time interval considered.

Let us assume now that S_t, I_t and O_t denote the values of the storage, inflow and outflow of that reach at time t, respectively. We have that:

$$\frac{dS_t}{dt} = I_t - O_t \tag{1}$$

In its linear form, the Muskingum model is given by:

$$S_t = K[\chi I_t + (1-\chi)O_t] \tag{2}$$

where K is a storage time constant for the river reach that approximates the flow travel time through the river reach, and χ is a weight usually taken values between 0 and 0.3 for a river or stream channel. This model is very simple, as it only depends on two parameters K and χ, which can be estimated from the inflow and outflow data. If the data on the inflow and outflow are sparse, then K can be safely taken as the travel time in the reach, while χ can be assumed to have an average value (the value 0.2 is commonly accepted). If hydrographs for the inflow and the outflow are available, a procedure decribed by McCarthy is to plot the discharge term $[\chi I_t + (1-\chi)O_t]$ against the accumulated storage for different values of the parameter χ and then select the value of χ generating the narrowest loop. However, this procedure is subjective, prone to errors and time consuming. The performance of this method can be improved by considering the nonlinear effects, as the relationship between the discharge term and the storage is not generally linear.

The nonlinear Muskingum model is given by the following evolution equations:

$$S_t = K[\chi I_t + (1-\chi)O_t]^m \tag{3}$$

where the new parameter m is an exponent introduced to account for the effects of nonlinearity.

In spite of its simplicity, the Muskingum model is still affected by a critical problem: the estimation of its parameters. Finding the correct values for K, χ and m is challenging, because such values cannot be obtained from the historical inflow an outflow hydrographs. This means that a powerful parameter estimation method is absolutely required. As a result, several optimization techniques have already been applied during the last two decades to tackle this issue. Early methods for this problem described in the literature include least-squares method (LSM) [6], Hook-Jeeves pattern search with linear regression, the conjugate gradient and Davidon-Fletcher-Powell method [11], and nonlinear least-squares regression [18]. More recently, two approximate methods based on computing the slopes of the inflow and outflow hydrographs at their intersection point and the computation of such hydrographs at two specific points were described in [1]. Other approach using the Broyden-Fletcher-Goldfard-Shanno (BFGS) method was reported in [5]. The method in [2] is based on the Nelder-mead simplex algorithm. Neither of these methods do guarantee the global optima. Recently, researchers in the field turned their attention towards nature-inspired metaheuristics. They include the use of genetic algorithms [9] and hybrid methods, such as a combination of the modified honey bee colony optimization with generalized reduced gradient methods in [10].

Among the myriad of nature-inspired metaheuristic methods for optimization, those based on swarm intelligence are receiving increasing attention during the las few years because of their ability to cope with problems where little or no information at all is available about the problem. These methods are also very effective for optimization problems where the objective function is not differentiable making gradient-based methods unsuitable, or for problems under difficult conditions (e.g., noisy data, irregular sampling) commonly found in real-world applications. Swarm intelligence methods applied to this problem include harmony search [7], and particle swarm optimization [4], artificial bee colony [12]. A very recent method also considers particle swarm optimization for this problem and investigates the effect of using variable values for the parameters [3].

It is worthwhile to remark that some of the previous methods do not consider the same Muskingum model addressed in this work, but other variations of it. For instance, the method in [3] consider the (simpler) linear Muskingum model, while the method in [1], although also nonlinear, considers a variation of Eq. (3) given by: $S_t = K[\chi I_t^n + (1-\chi)O_t^n]$. In other words, there are several (qualitatively different) formulations of the Muskingum model so it is important to notice which one is actually under analysis. This observation is crucial to avoid leading our readers to confusion about the real model used in this contribution and its possible relationship with other previous approaches.

In this paper, we address the parameter estimation problem for the nonlinear Muskingum method given by Eqs. (1)-(3). Our approach is based a powerful nature-inspired swarm intelligence technique for global optimization called bat algorithm. The structure of this paper is as follows: the main steps of the procedure for the nonlinear Muskingum model are briefly described in Sect. 2. The bat algorithm is described in detail in Sect. 3 and then applied to our problem in Sect. 4. Our experimental results are briefly discussed in Sect. 6. The paper closes with the main conclusions and some ideas for future work in the field.

2 Procedure for the Nonlinear Muskingum Model

Rearranging Eq. (3), the rate of outflow becomes:

$$O_t = \left(\frac{1}{1-X}\right)\left(\frac{S_t}{K}\right)^{\frac{1}{m}} - \left(\frac{X}{1-X}\right) I_t \tag{4}$$

Combining Eqs. (1) and Eq. (4), we get:

$$\frac{\Delta S_t}{\Delta t} = \left(\frac{1}{1-X}\right)\left(\frac{S_t}{K}\right)^{\frac{1}{m}} - \left(\frac{X}{1-X}\right) I_t \tag{5}$$

$$S_{t+1} = S_t + \Delta S_t \tag{6}$$

$$O_{t+1} = \left(\frac{1}{1-X}\right)\left(\frac{S_{t+1}}{K}\right)^{\frac{1}{m}} - \left(\frac{X}{1-X}\right) \bar{I}_{t+1} \tag{7}$$

where $\bar{I}_{t+1} = (I_{t+1} + I_t)/2$. Then, the procedure for the nonlinear Muskingum model given by Eqs. (1)-(3) is based on the following steps:

- **Step 1:** Assume initial values for the three parameters K, χ and m.
- **Step 2:** Calculate the storage S_t using Eq. (3), taking the initial outflow equal to the initial inflow.
- **Step 3:** Calculate the time rate of storage using Eq. (5).
- **Step 4:** Estimate the next accumulated storage using Eq. (6).
- **Step 5:** Calculate the next outflow using Eq. S_t using Eq. (7). I_t will replace $\bar{I}_{t+1}$ when the ratio of storage t and $t+1$ exceeds 2.
- **Step 6:** Repeat the steps 2-5.

3 The Bat Algorithm

The *bat algorithm* is a nature-inspired swarm intelligence algorithm proposed by X.S. Yang in 2010 to solve optimization problems [15–17]. The algorithm is inspired by the echolocation behavior of bats, which is used as a metaphor for a global optimization method and is described by three idealized rules:

1. Bats use echolocation to sense distance and distinguish between food, prey and background barriers.
2. Each virtual bat flies randomly with a velocity $\mathbf{v}_i$ at position (solution) $\mathbf{x}_i$ with a fixed frequency f_{min}, varying wavelength λ and loudness A_0 to search for prey. As it searches and finds its prey, it changes wavelength (or frequency) of their emitted pulses and adjust the rate of pulse emission r, depending on the proximity of the target.
3. It is assumed that the loudness will vary from an (initially large and positive) value A_0 to a minimum constant value A_{min}.

```
Require: (Initial Parameters)
    Population size: $\mathcal{P}$                Maximum number of generations: $\mathcal{G}_{max}$
    Loudness: $\mathcal{A}$                       Pulse rate: $r$
    Maximum frequency: $f_{max}$                  Dimension of the problem: $d$
    Objective function: $\phi(\mathbf{x})$, with $\mathbf{x} = (x_1, \ldots, x_d)^T$
    Random number: $\theta \in U(0,1)$
 1: $g \leftarrow 0$
 2: Initialize the bat population $\mathbf{x}_i$ and $\mathbf{v}_i$, $(i = 1, \ldots, n)$
 3: Define pulse frequency $f_i$ at $\mathbf{x}_i$
 4: Initialize pulse rates $r_i$ and loudness $\mathcal{A}_i$
 5: while $g < \mathcal{G}_{max}$ do
 6:    for $i = 1$ to $\mathcal{P}$ do
 7:       Generate new solutions by adjusting frequency,
 8:       and updating velocities and locations //eqns. (8)-(10)
 9:       if $\theta > r_i$ then
10:          $\mathbf{s}^{best} \leftarrow \mathbf{s}^g$        //select the best current solution
11:          $\mathbf{ls}^{best} \leftarrow \mathbf{ls}^g$     //generate a local solution around $\mathbf{s}^{best}$
12:       end if
13:       Generate a new solution by local random walk
14:       if $\theta < \mathcal{A}_i$ and $\phi(\mathbf{x_i}) < \phi(\mathbf{x}^*)$ then
15:          Accept new solutions
16:          Increase $r_i$ and decrease $\mathcal{A}_i$
17:       end if
18:    end for
19:    $g \leftarrow g + 1$
20: end while
21: Rank the bats and find current best $\mathbf{x}^*$
22: return $\mathbf{x}^*$
```

Algorithm 1: Bat algorithm pseudocode

In general, we assume that the frequency f evolves on a bounded interval $[f_{min}, f_{max}]$. For simplicity, we can assume that $f_{min} = 0$, so $f \in [0, f_{max}]$. The rate of pulse can simply be in the range $r \in [0,1]$, where 0 means no pulses at all, and 1 means the maximum rate of pulse emission. The pseudocode of the algorithm is shown in Algorithm 1. Basically, it considers an initial population of $\mathcal{P}$ individuals (bats). Each bat, representing a potential solution, has a location $\mathbf{x}_i$ and velocity $\mathbf{v}_i$, initialized with random values within the search space. Then, the pulse frequency, pulse rate, and loudness are computed for each individual bat. The swarm evolves iteratively over generations until the maximum number of generations, $\mathcal{G}_{max}$, is reached. For each generation g and each bat, new frequency, location and velocity are computed as:

$$f_i^g = f_{min}^g + \beta(f_{max}^g - f_{min}^g) \tag{8}$$

$$\mathbf{v}_i^g = \mathbf{v}_i^{g-1} + [\mathbf{x}_i^{g-1} - \mathbf{x}^*] \, f_i^g \tag{9}$$

$$\mathbf{x}_i^g = \mathbf{x}_i^{g-1} + \mathbf{v}_i^g \tag{10}$$

where $\beta \in [0,1]$ follows the random uniform distribution, and $\mathbf{x}^*$ represents the current global best location (solution), which is obtained through evaluation of the objective function at all bats and ranking of their fitness values. The superscript $(.)^g$ is used to denote the current generation g.

The best current solution and a local solution around it are probabilistically selected and the search is intensified by a local random walk. For this local search, the solution selected is perturbed locally through a random walk as. $\mathbf{x}_{new} = \mathbf{x}_{old} + \epsilon \mathcal{A}^g$, where ϵ is a uniform random number on the interval $[-1,1]$ and $\mathcal{A}^g = < \mathcal{A}_i^g >$, is the average loudness of all the bats at generation g.

If the new solution is better than the previous best one, it is probabilistically accepted depending on the value of the loudness. In that case, the algorithm increases the pulse rate and decreases the loudness. This process is repeated for the given number of generations. In general, the loudness decreases once a bat finds its prey (in our analogy, once a new best solution is found), while the rate of pulse emission decreases. For simplicity, the following values are commonly used: $\mathcal{A}_0 = 1$ and $\mathcal{A}_{min} = 0$, assuming that this latter value means that a bat has found the prey and temporarily stop emitting any sound. The evolution rules for loudness and pulse rate are: $\mathcal{A}_i^{g+1} = \alpha \mathcal{A}_i^g$ and $r_i^{g+1} = r_i^0[1 - exp(-\gamma g)]$, where α and γ are constants. Note that for any $0 < \alpha < 1$ an any $\gamma > 0$ we have: $\mathcal{A}_i^g \to 0, \quad r_i^g \to r_i^0, \quad \text{as} \quad g \to \infty$. In general, each bat should have different values for loudness and pulse emission rate. To this aim, we can take an initial loudness $\mathcal{A}_i^0 \in (0,2)$ while the initial emission rate r_i^0 can be any value in the interval $[0,1]$. Loudness and emission rates will be updated only if the new solutions are improved, an indication that the bats are moving towards the optimal solution. As a result, the bat algorithm applies a parameter tuning technique to control the dynamic behavior of a swarm of bats. Similarly, the balance between exploration and exploitation can be controlled by tuning algorithm-dependent parameters.

4 The Method

To apply bat algorithm described above to our problem, we need to define some important issues. Firstly, we need an adequate representation of the problem. Each bat $\mathcal{B}_j$, representing a potential solution, corresponds to a parametric vector of the free variables of the problem, in the form: $\mathcal{B}_j = (K_j, \chi_j, m_j)$. These parametric vectors are initialized with random values on the intervals $(0,10)$ for K and m and $(0,5)$ for χ. We remark however that these constraints apply only for the initial conditions, as we allow the variables to move freely outside such ranges during the execution of the algorithm. Secondly, a fitness function is required for optimization. In our problem, the goal is to predict the outflow given the inflow and then compare the predicted outflow with the observed one. This can be properly done through least-squares minimization. Let O_t an $\bar{O}_t$ be the observed and the predicted outflow at time t, respectively. We consider the least-squares functional LSQ given by the sum of the squares of the residuals:

$$\text{Minimize}(LSQ) = \underset{\{K_j\},\{\chi_j\},\{m_j\}}{\text{Minimize}} \left[\sum_{t=1}^{n} (O_t - \bar{O}_t)^2 \right] \tag{11}$$

Table 1. Observed inflow and outflow hydrographs and computed outflow of a state-of-the-art PSO method and of our bat algorithm method for the example in the paper (best results are highlighted in bold)

		Observed	*Computed*
Time (h)	*Inflow (cms)*	*Outflow (cms)*	*Our Method*
0	105	108	108.4
1	106	108	108.2
2	107	109	109.1
3	107	109	109.7
4	108	109	109.2
5	109	109	108.8
6	110	110	109.9
7	110	110	110.0
8	111	110	109.8
9	111	110	110.2
10	111	110	109.9
11	114	111	111.0
12	117	112	111.9
13	120	112	112.3
14	122	112	112.1
15	141	112	111.9
16	160	112	112.3
17	177	113	112.9
18	201	113	113.3
19	227	114	114.1
20	258	116	116.5
21	295	119	118.8
22	340	122	121.9
23	392	125	125.0
24	446	131	131.1
25	505	13	137.2

where n denotes the number of time instances of the inflow/outflow time series. Finally, we need to address the important issue of parameter tuning. It is well-known that the performance of swarm intelligence techniques depends of a proper parameter tuning, which is also problem-dependent. Due to this reason, our choice has been fully empirical, based on numerous computer simulations for different parameter values. In this paper, we consider a population size of 30 as larger population sizes do increase the CPU times without significantly improving our numerical results. The initial and minimum loudness and parameter α are set to 0.5, 0, and 0.2, respectively. Regarding the stopping criterion, all executions are performed until no further improvement is achieved after 20 consecutive generations.

Table 2. Parameter values and LSQ error obtained for the methods in our comparison (best error highlighted in bold).

Method	*SI approach*	K	χ	m	LSQ
Chu & Chang [4]	PSO	0.1824	0.3330	2.1458	36.89
Our method	Bat algorithm	3.4580	0.0034	2.3065	**19.59**

5 Experimental Results

Our method has been tested on an illustrative example first proposed in 1974 by Wilson [13] and corresponding to a channel routing problem. This example has been widely used as a benchmark for different methods in several previous works. Table 1 reports the values of the observed inflow and outflow (columns 2 and 3) for different time instances, expressed in hours (column 1). All flow results are expressed in cubic meters per second (cms). We also report the numerical results of a previous method reported in [4] and based on PSO (column 4) and the results of our method based on the bat algorithm (column 5). To avoid the spurious effects derived from the randomness of the process, we run 15 independent executions of our method and then consider the average value. This means that the results reported in the last column are not those of the best execution (which are even better) but the average of the 15 independent executions. Still, the numerical results obtained with our method are very relevant. A visual comparison between the columns 3 and 5 reveals that our method performs very well, as it is able to capture the real tendency of data for all times instances in the example. This means that our method has a very good predictive capability.

We also compare our results with those of a method based on PSO and reported in [4]. This method has been primarily chosen for comparison because it is widely considered a state-of-the-art method in the field and outperforms many other methods in the literature for this problem. The best result for each time instance has been highlighted in bold for easier comparison. As the reader can see, our method outperforms that in [4] for most time instances in this example. This fact becomes evident from Table 2, where we show the optimal parameter values (columns 3-5) along with the LSQ error (column 6) obtained with both methods (described in columns 1-2). Once again, the best result is in bold. Note that our method improves the LSQ of the PSO method significantly.

Our good numerical results are confirmed visually in Figure 1. The figure depicts the observed inflow and outflow hydrographs as well as the predicted outflow for the PSO and bat algorithm methods used in our comparative work. Note the excellent visual matching between the observed outflow and the outflow predicted by our method. Although the outflow of the PSO method is very close to the observed one, ours is even better as it becomes visually indistinguishable from the observed outflow and they overlap each other. This is the best indicator of the good performance of our method for this real-world example.

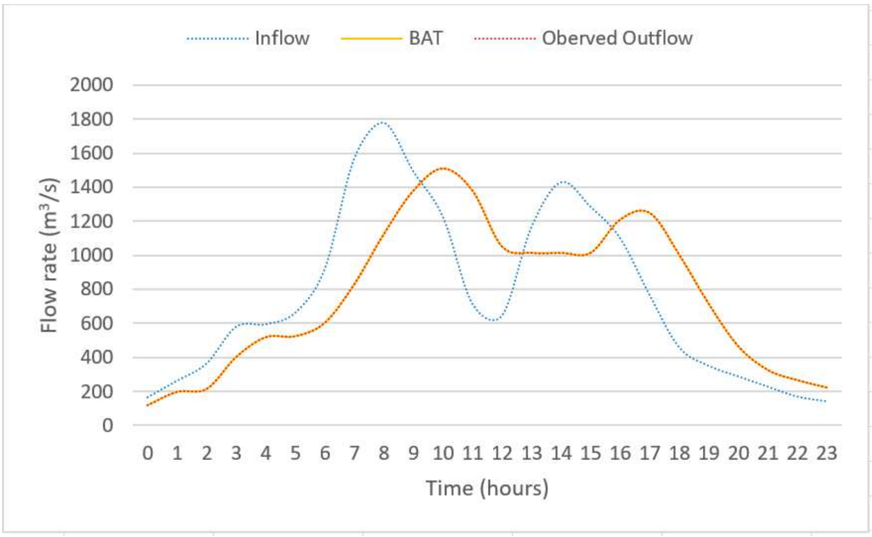

Fig. 1. Observed and predicted inflow and outflow hydrographs for the example computed with the parameter values obtained with PSO and our bat algorithm method.

Finally, Fig. 2 shows the convergence diagram of our method for the maximum, minimum and average values from the 15 independent executions. As shown, the method converges in all cases, and there is no large variation between the different executions, meaning that the method is robust for different executions, a very valuable feature for real-world applications.

6 Conclusions and Future Work

In this paper we present a bat algorithm-based method to solve the parameter estimation problem of the nonlinear Muskingum model, a relevant problem in hydrology, flood forecasting, dam design and other engineering fields. The method is simple to understand and easy to implement. Our computational experiments for a popular real-world channel routing example used as a benchmark in the field show that it performs very well, is robust and outperforms other approaches in the field. We conclude that it can be safely used for outflow prediction in flood forecasting and in other tasks.

Future work in the field includes the extension of this approach to the case of natural rivers and reservoir routing, for which the parameters are expected to behave differently. Improving the accuracy of our method even further is also part of our future work in the field.

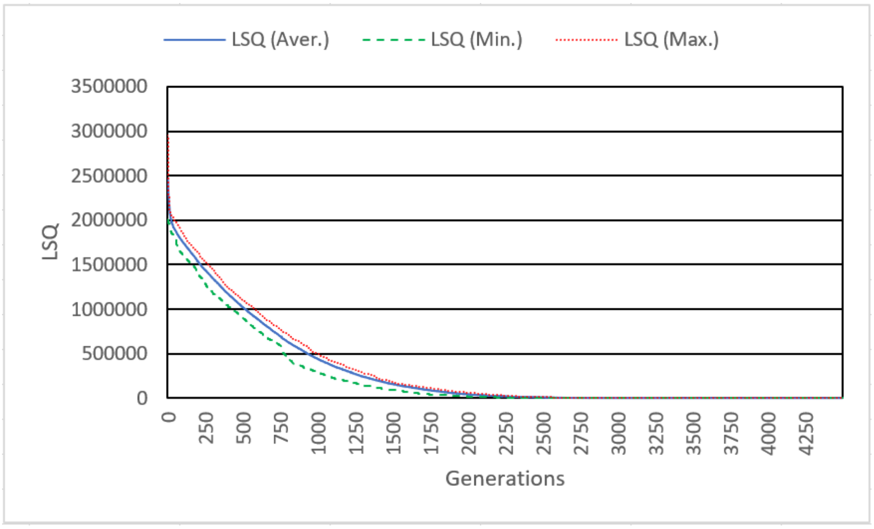

Fig. 2. Convergence diagram of the PSQ error function for the minimum, maximum and average values for 15 different executions.

Acknowledgements

Research supported by: project PDE-GIR of the EU Horizon 2020 research and innovation program, Marie Sklodowska-Curie grant agreement No 778035; project #TIN2017-89275-R of Agencia Estatal de Investigación and EU Funds FEDER (AEI/FEDER-UE); and project #JU12, of SODERCAN and EU Funds FEDER (SODERCAN/FEDER-UE).

References

1. Al-Humoud, J. M., Esen, I. I.: Approximate methods for the estimation of Muskingum flood routing parameters. *Water Resources Management*, **20**, 979–990 (2006).
2. Barati, R: Parameter estimation of nonlinear Muskingum models using Nelder-Mead simplex algorithm. *J Hydrol Eng.*, **16**(11), 946954 (2011).
3. Bazargan, J., Norouzi, H: Investigation the effect of using variable values for the parameters of the linear Muskingum method using the particle swarm algorithm (PSO). *Water Resources Management*, **32**(14), 47634777 (2018).
4. Chu, H.J., Chang, L.C.: Applying particle swarm optimization to parameter estimation of the nonlinear Muskingum model. J Hydrol Eng 14(9):10241027 (2009).
5. Geem, Z. W.: Parameter estimation for the nonlinear Muskingum model using the BFGS technique. *J. Irrig. Drain. Eng.*, **1325**, 474–478 (2006).
6. Gill, M. A.: Flood routing by Muskingum method. *J. Hydrol.*, **363–4**, 353–363 (1978).
7. Kim, J. H., Geem, Z. W., Kim, E. S.: Parameter estimation of the nonlinear Muskingum model using harmony search. *J. Am. Water Resour. Assoc.*, **375**, 1131–1138 (2001).

8. McCarthy G. T.: The unit hydrograph and flood routing. New London. *Conference North Atlantic Division. US Army Corps of Engineers*. New London. Conn. USA (1938).
9. Mohan, S.: Parameter estimation of nonlinear Muskingum models using genetic algorithm. *J. Hydraul. Eng.*, **1232**, 137–142 (1997).
10. Niazkar, M., Afzali, S.H.: Application of new hybrid optimization technique for parameter estimation of new improved version of Muskingum model. *Water Resources Management* **30**(13):47134730 (2016).
11. Tung, Y. K.: River flood routing by nonlinear Muskingum method. *J. Hydraul. Eng.*, **111**(12), 1447–1460 (1985).
12. Vafakhah, M., Dastorani, A., Moghaddam, A.: Optimal parameter estimation for nonlinear Muskingum model based on artificial bee Colony algorithm. *EcoPersia*, **3**(1):847865 (2015).
13. Wilson, E. M.: *Engineering Hydrology*, MacMillan, Hampshire, U.K. (1974).
14. Yang, X.-S.: *Nature-Inspired Metaheuristic Algorithms (2nd. Edition).* Luniver Press, Frome, UK (2010).
15. Yang, X.S.: A new metaheuristic bat-inspired algorithm. *Studies in Computational Intelligence*, Springer Berlin, **284**, 65–74 (2010).
16. Yang, X. S..: Bat algorithm for multiobjective optimization. *Int. J. Bio-Inspired Computation*, **3**(5) (2011), 267-274.
17. Yang, X.S., Gandomi, A.H.: Bat algorithm: a novel approach for global engineering optimization. *Engineering Computations*, **29**(5) (2012), 464–483.
18. Yoon, J. W., Padmanabhan, G.: Parameter-estimation of linear and nonlinear Muskingum models. *J. Water Resour. Plann. Manage.*, **1195**, 600–610 (1993).