Parameter Estimation of a Nonlinear Hydrologic Model for Channel Flood Routing with the Bat Algorithm

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. Flood routing is a methodology to predict the changes of the flow of water as it moves through a natural river, an artificial channel, or a reservoir. It is widely used in fields such as flood prediction, reservoir design, geographic planning, and many others. One of the most popular and widely used flood routing techniques is the Muskingum model, as it is conceptually simple and only depends on a few parameters that can be estimated from historical inflow/outflow records. However, the estimation of such parameters for the nonlinear case is still a challenging task. In this paper we present a method based on a powerful swarm intelligence technique called bat algorithm to solve the parameter estimation problem of the nonlinear Muskingum model for channel routing. The method is applied to an illustrative example used as a benchmark in the field with very good results. We also show that our method outperforms other state-of-the-art methods in the field such as PSO.

Keywords: Swarm intelligence · global optimization · bat algorithm · flood routing · hydrologic models · parameter estimation

1 Introduction

Flood routing is a very important subject of research in water engineering and other fields. Roughly speaking, flood routing aims at predicting the changes of the rate of flow (discharge) over the time for a given section or point of a river, channel, or reservoir. This problem has many relevant applications in several areas, such as hydrology, flood forecasting, watershed simulations, geographic planning, flood protection, reservoir design, and many others.

Several methods have been described in the literature to address this problem. Among them, the Muskingum method is one of the most popular and frequently used methods for engineering of rivers and channels owing to its simplicity. This method, originally proposed in 1938 for the Muskingum river in Ohio [10], is indeed very simple to use, as it only depends on two parameters that can be estimated from past inflow and outflow data. The nonlinear Muskingum model is given by the following evolution equations:

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

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

where S_t, I_t and O_t denote the values of the storage, inflow and outflow at time t, respectively, K is a storage time constant for the river reach that provides an approximates the flow travel time through the river reach, χ is a weight usually taken values between 0 and 0.3 for stream channels, and 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 and 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 [13], and nonlinear least-squares regression [20]. 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 [11] and hybrid methods, such as a combination of the modified honey bee colony optimization with generalized reduced gradient methods in [12].

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 [9], and particle swarm optimization [4], artificial bee colony [14]. 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. (2) 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)-(2). 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. 5. 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. (2), the rate of outflow becomes:

$$O_t = \left(\frac{1}{1-X}\right)\left(\frac{S_t}{K}\right)^{\frac{1}{m}} - \left(\frac{X}{1-X}\right) I_t \tag{3}$$

Combining Eqs. (1) and Eq. (3), we get:

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

$$S_{t+1} = S_t + \Delta S_t \tag{5}$$

$$O_{t+1} = \left(\frac{1}{1-X}\right)\left(\frac{S_{t+1}}{K}\right)^{\frac{1}{m}} - \left(\frac{X}{1-X}\right) \bar{I}_{t+1} \tag{6}$$

where $\bar{I}_{t+1} = (I_{t+1} + I_t)/2$. Then, the procedure for the nonlinear Muskingum model given by Eqs. (1)-(2) 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. (2), taking the initial outflow equal to the initial inflow.
- **Step 3:** Calculate the time rate of storage using Eq. (4).
- **Step 4:** Estimate the next accumulated storage using Eq. (5).
- **Step 5:** Calculate the next outflow using Eq. S_t using Eq. (6). 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 [17–19]. 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 a (initially large and positive) value A_0 to a minimum constant value A_{min}.

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 pseudo-code 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:

$$f_i^g = f_{min}^g + \beta(f_{max}^g - f_{min}^g) \tag{7}$$

$$\mathbf{v}_i^g = \mathbf{v}_i^{g-1} + \left[\mathbf{x}_i^{g-1} - \mathbf{x}^*\right] f_i^g \tag{8}$$

$$\mathbf{x}_i^g = \mathbf{x}_i^{g-1} + \mathbf{v}_i^g \tag{9}$$

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

```
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 $\mathbf{x} = (x_1, \dots, x_d)^T$
    Random number: $\theta \in U(0,1)$
 1: $g \leftarrow 0$
 2: Initialize the bat population $\mathbf{x}_i$ and $\mathbf{v}_i$, $(i = 1, \dots, n)$
 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. (7)-(9)
 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

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 $0 < \alpha < 1$ and any $\gamma > 0$ we have: $\mathcal{A}_i^g \to 0, \quad r_i^g \to r_i^0, \quad \text{as} \ \ g \to \infty$. 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

Bat algorithm has already been applied to data fitting problems [7, 8]. Still, 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 and $\bar{O}_t$ 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:

$$\text{Minimize}(LSQ) = \underset{\{K_j\},\{\chi_j\},\{m_j\}}{\text{Minimize}} \left[\sum_{t=1}^{n} (O_t - \bar{O}_t)^2 \right] \tag{10}$$

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.

5 Experimental Results

Our method has been tested on an illustrative example first proposed in 1974 by Wilson [15] 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 two previous methods, reported in [11] and [4] and based on genetic algorithms (GA) and PSO respectively, (columns 4 and 5) and the results of our method based on the bat algorithm (column 6). 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

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

		Observed	*Computed Outflow (cms)*		
Time (h)	*Inflow (cms)*	*Outflow (cms)*	*GA*	*PSO*	*Our Method*
0	22	22	**22.0**	**22.0**	23.0
6	23	21	22.0	22.0	**21.5**
12	35	21	22.4	22.6	**22.2**
18	71	26	26.3	28.1	**26.2**
24	103	34	34.2	32.2	**34.0**
30	111	44	**44.2**	45.0	43.3
36	109	55	56.9	57.0	**55.2**
42	100	66	68.2	67.5	**65.6**
48	86	75	77.1	75.9	**74.4**
54	71	82	83.2	**81.2**	81.0
60	59	85	85.7	**85.6**	**84.4**
66	47	84	**84.2**	**84.2**	83.5
72	39	80	80.2	79.6	**79.9**
78	32	73	**73.3**	**73.3**	72.5
84	28	64	65.0	65.0	**63.6**
90	24	54	55.8	56.2	**53.8**
96	22	44	46.7	46.5	**45.1**
102	21	36	38.0	37.3	**36.5**
108	20	30	30.9	**29.7**	30.4
114	19	25	25.7	24.3	**25.5**
120	19	22	**22.1**	20.6	22.7
126	18	19	20.4	**19.6**	20.0

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

Method	*SI approach*	K	χ	m	*LSQ*	*I.R.*
Mohan [11]	Genetic algorithms	0.1033	0.2873	1.8282	38.23	0.00%
Chu & Chang [4]	Particle swarm optimization	0.1824	0.3330	2.1458	36.89	3.50%
Our method	Bat algorithm	3.4580	0.0034	2.3065	**19.59**	**48.78%**

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 two methods based on GA [11] and PSO [4]. These methods have been primarily chosen for comparison because

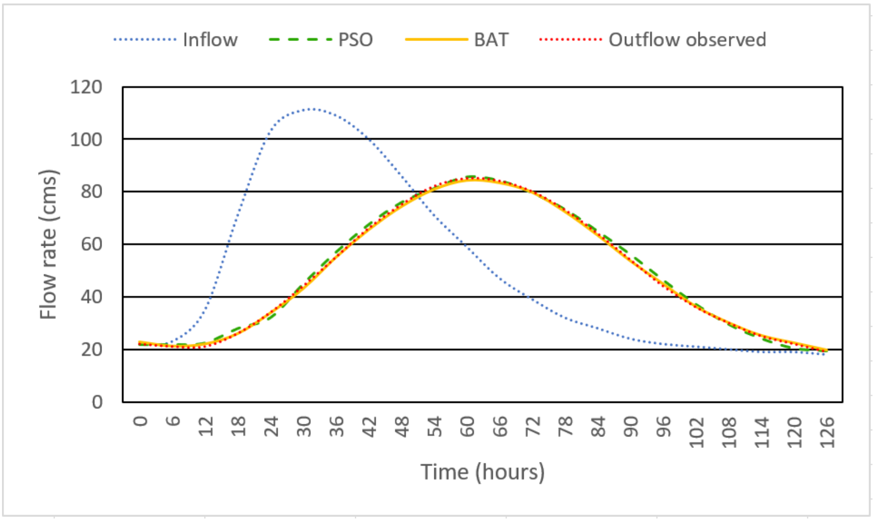

Fig. 1. Observed and predicted inflow and outflow hydrographs for the example computed with the parameter values obtained with PSO and our bat algorithm method.

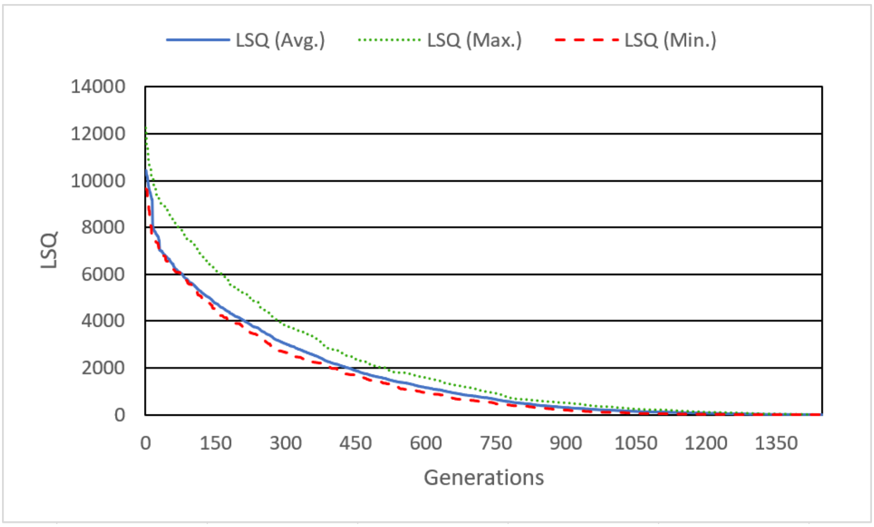

Fig. 2. Convergence diagram of the LSQ error function for the minimum, maximum and average values for 15 different executions.

they are very popular and widely considered state-of-the-art methods in the field and outperform 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 these methods 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 the three methods (described in columns 1-2), and the improvement rate (column 7) obtained with respect to the worst method (the GA, in this case). Once again, the best results are in bold. Note that our method improves the LSQ error of the GA and PSO methods significantly, with an improvement rate of 48.78% and 46.89% over GA and PSO, respectively.

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.

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 state-of-the-art approaches in the field. We conclude that it can be safely used for outflow prediction in flood forecasting and in related 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 quite differently. Improving the accuracy of our method even further is also part of our future work in the field.

Acknowledgements

The research in this paper has been supported by the project PDE-GIR of the EU Horizon 2020 research and innovation program, Marie Sklodowska-Curie grant agreement No 778035; the project #TIN2017-89275-R of the Agencia Estatal de Investigación (Spanish Ministry of Science, Innovation and Universities) and EU Funds EFRD (AEI/FEDER-UE); and the project #JU12, of SODERCAN and

EU Funds EFRD (SODERCAN /FEDER-UE). We also thank the reviewers for their insightful comments and suggestions.

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), 946–954 (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), 4763–4777 (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):1024–1027 (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. Iglesias, A., Gálvez, A., Collantes, M.: Multilayer embedded bat algorithm for B-spline curve reconstruction. *Int. Computer-Aided Eng.*, **24**(4), 385–399 (2017).
8. Iglesias, A., Gálvez, A., Collantes, M.: Iterative sequential bat algorithm for free-form rational Bézier surface reconstruction. *Int. J. of Bio-Inspired Computation*, **11**(1), 1–15 (2018).
9. 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).
10. McCarthy G. T.: The unit hydrograph and flood routing. New London. *Conf. North Atlantic Division. US Army Corps of Engineers*. New London. Conn. USA (1938).
11. Mohan, S.: Parameter estimation of nonlinear Muskingum models using genetic algorithm. *J. Hydraul. Eng.*, **1232**, 137–142 (1997).
12. 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):4713–4730 (2016).
13. Tung, Y. K.: River flood routing by nonlinear Muskingum method. *J. Hydraul. Eng.*, **111**(12), 1447–1460 (1985).
14. Vafakhah, M., Dastorani, A., Moghaddam, A.: Optimal parameter estimation for nonlinear Muskingum model based on artificial bee Colony algorithm. *EcoPersia*, **3**(1):847–865 (2015).
15. Wilson, E. M.: *Engineering Hydrology*, MacMillan, Hampshire, U.K. (1974).
16. Yang, X.-S.: *Nature-Inspired Metaheuristic Algorithms (2nd. Edition)*. Luniver Press, Frome, UK (2010).
17. Yang, X.S.: A new metaheuristic bat-inspired algorithm. *Studies in Computational Intelligence*, Springer Berlin, **284**, 65–74 (2010).
18. Yang, X. S..: Bat algorithm for multiobjective optimization. *Int. J. Bio-Inspired Computation*, **3**(5) (2011), 267-274.
19. Yang, X.S., Gandomi, A.H.: Bat algorithm: a novel approach for global engineering optimization. *Engineering Computations*, **29**(5) (2012), 464–483.
20. Yoon, J. W., Padmanabhan, G.: Parameter-estimation of linear and nonlinear Muskingum models. *J. Water Resour. Plann. Manage.*, **1195**, 600–610 (1993).