# Global-Support Rational Curve Method for Data Approximation with Bat Algorithm

Andrés Iglesias, Akemi Gálvez, Marta Collantes

# Global-Support Rational Curve Method for Data Approximation with Bat Algorithm

Andrés Iglesias[1,2,†], Akemi Gálvez[1], Marta Collantes[1]

[1]Department of Applied Mathematics and Computational Sciences,
University of Cantabria, Avda. de los Castros, s/n, E-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 problem of obtaining an approximating curve from a given set of data points appears recurrently in several applied and industrial domains, such as CAD/CAM, computer graphics and animation, medicine, and many others. Although polynomial blending functions are usually applied to tackle this issue, some shapes cannot yet be adequately approximated by using the polynomial scheme. In this paper we address this limitation by applying rational global-support blending functions, particularly rational Bézier curves. Our method is based on a nature-inspired meta-heuristic called bat algorithm, which has been recently introduced to solve difficult optimization problems. To check the performance of our approach, it has been applied to some illustrative examples of 2D and 3D curves. Our results show that the method performs very well, being able to yield a satisfactory approximating curve with a high degree of accuracy.

**Keywords:** data approximation; rational curve; bat algorithm; meta-heuristic technique; Bézier curve

## 1 Introduction

### 1.1 Motivation

Obtaining an approximating curve from a given set of data points is a very important and recurrent problem in many applied and industrial domains. It appears, for instance, in computer-aided design and manufacturing (CAD/CAM), a field where data points are usually obtained from real measurements of an existing geometric entity, as it typically happens in the construction of car bodies, ship hulls, airplane fuselage, and other free-form objects. It is also a common problem in the shoes industry, in archeology (reconstruction of archeological assets), in medicine (computer tomography (CT), magnetic resonance imaging), computer graphics and animation, virtual and augmented reality, and in many other fields.

Data points in these settings are acquired through many different technologies, such as 3D laser scanning, touch scanners, coordinate measuring machines, CT scanners, or convergent photogrammetry, to mention just a few. A common factor of all these techniques is that a huge number of data (often in the range of hundreds of thousands, and even millions) is usually obtained. Besides, in most cases no geometric or topological information is available beyond the data points. Since dealing with this amount of data becomes impractical, a primary goal in the field is to convert the real data from a physical object into a fully usable digital model, a process commonly called *reverse engineering*. This allows significant savings in terms of storage capacity and processing and manufacturing time. Furthermore, the digital models are easier and cheaper to modify than their real counterparts and are usually available anytime and anywhere.

In real-world problems, data points are usually affected by measurement noise, irregular sampling, and other artifacts [2, 27, 28]. Consequently, a good fitting of data should be generally based on approximation schemes rather than on interpolation. In this case, the approximating curve is not required to pass through all input data points, but just near to them, according to some prescribed distance criteria.

Several approximating families of functions have been applied to this problem. Among them, the free-form parametric curves such as Bézier, B-spline and NURBS, are widely applied in many industrial settings due to their great flexibility and the fact that they can represent smooth shapes with only a few parameters [2, 21, 22, 25, 26]. In general, the approximating curves can be classified as global-support and local-support. By global-support curves we mean curves mathematically expressed as a combination of basis functions whose support is the whole domain of the problem. As a consequence, these curves exhibit a global control, in the sense that any modification of the shape of the curve in a particular location is propagated throughout the whole curve. This is in clear contrast to the local-support approaches, which provide local control of the shape of the curve [23, 28] and have become prevalent in CAD/CAM and computer graphics. In this work we focus on the global-support approach.

Some previous papers addressed this problem by using Bézier curves [12, 23], which are given by a linear combination of polynomial basis functions (the Bernstein polynomials). Although they obtained good results for a number of shapes, this polynomial approach is still limited, as it cannot adequately describe some particular shapes (such as the conics). As a consequence, there is still a need for more powerful blending functions.

An interesting extension in this regard is given by the rational basis functions, which are mathematically described as the quotient of two polynomials. A remarkable advantage of this rational scheme is that the conics can be canonically described as rational functions. In this paper, we take advantage of this valuable feature to solve the curve approximation problem by using rational Bézier curves. Unfortunately, this rational approach becomes more difficult than the polynomial one, since new parameters are now introduced into the problem. Consequently, we are confronted with the challenge of obtaining optimal values for many (qualitatively different) parameters, namely, data parameters, poles, and weights. This leads to a difficult over-determined multivariate continuous nonlinear optimization problem.

### 1.2 Aims and Structure of the Paper

In this paper, we solve this optimization problem by applying a recently proposed nature-inspired meta-heuristic technique called *bat algorithm*, which is receiving increasing attention from the scientific community during the last few years due to its good performance in solving difficult continuous optimization problems [33]. This algorithm is based on the echolocation behavior of microbats with varying pulse rates of emission and loudness (see Section 3 for details). We would like to remark that, in spite of its valuable features for continuous optimization, to the best of authors' knowledge the bat algorithm has never been applied in the context of data fitting for geometric modeling or computer graphics. The present paper aims to fill this gap. In this work, the bat algorithm is applied to obtain a very accurate fitting curve to a given set of data points by using rational global-support blending functions. To check the performance of our approach, it has been applied to three simple yet illustrative examples of two-dimensional and three-dimensional shapes.

The structure of this paper is as follows: previous work in the subject of data fitting with free-form parametric curves is briefly reported in Section 2. In Section 3 we provide a gentle overview about the bat algorithm and its main features and advantages. The problem of curve approximation with rational Bézier curves along with our proposed approach to solve it are described in Section 4. The section also discusses the important issue of parameter tuning. Then, three illustrative examples of its application are reported in Section 5. Our experimental results show that the presented method performs very well, being able to replicate the underlying shape of data accurately. The paper closes with the main conclusions of this contribution and our plans for future work in the field.

## 2 Previous Work

The problem of data approximation with free-form parametric curves has been the subject of research for many years. First approaches in the field were mostly based on numerical procedures [3, 4, 29]. However, it has been shown that traditional mathematical optimization techniques fail to solve the problem in its generality. Consequently, there has been a great interest to explore other possible approaches to this problem. Some recent approaches in this line use error bounds [25], curvature-based squared distance minimization [32], or dominant points [26]. In general, they perform well but require some particular constraints (such as high differentiability, closed curves, noiseless data) which are not so commonly met in real-world applications.

On the other hand, interesting research carried out during the last two decades has shown that the application of artificial intelligence and soft computing techniques can achieve remarkable results for this problem [1, 17, 18]. Most of these methods rely on some kind of neural networks, such as standard neural networks [17], or Kohonen's SOM (Self-Organizing Maps) nets [18]. In some cases, this neural approach is combined with partial differential equations [1] or other approaches [20]. The generalization of these methods to functional networks is also analyzed in [5, 19]. The application of support vector machines to solve the least-squares B-spline curve fitting problem is reported in [21].

Other approaches are based on the application of nature-inspired metaheuristic techniques, which have been intensively applied to solve difficult optimization problems that cannot be tackled through traditional optimization algorithms. Genetic algorithms have been applied to this problem in both the discrete version [31] and the continuous version [16, 37]. Other metaheuristic approaches applied to this problem include the use of the popular particle swarm optimization technique [6, 7], artificial immune systems [14, 15], firefly algorithm [9–11], cuckoo search [12], simulated annealing [23], estimation of distribution algorithms [38], memetic algorithms [13], and hybrid techniques [8, 31]. These methods yield good results for shapes that can be properly described in polynomial terms, but more complicated shapes are still elusive. Furthermore, it has been shown that even some simple shapes such as the conics cannot be adequately described in terms of polynomial functions. In this paper we overcome this limitation by considering rational basis functions, as discussed in section 4.

## 3 The Bat Algorithm

### 3.1 Basic Principles

The bat algorithm is a bio-inspired population-based meta-heuristic algorithm originally proposed by Xin-She Yang in 2010 to solve difficult optimization problems [33, 35]. The algorithm is based on the echolocation behavior of bats. The author focused particularly on microbats, as they use a type of sonar called echolocation, with varying pulse rates of emission and loudness, to detect prey, avoid obstacles, and locate their roosting crevices in the dark.

Despite the short time since its appearance, the bat algorithm has already been applied to several engineering and industrial problems. Simultaneously, some modifications and improvements on the original version have been developed, such as the muti-objective bat algorithm [34], directed artificial bat algorithm [30], binary bat algorithm [24], and others. The interested reader is referred to the general paper in [36] for a comprehensive, updated review and taxonomic classification of the bat algorithm and all its variants and applications.

In this paper we consider the standard bat algorithm, as described in the original paper in [33]. According to that source, the idealization of the echolocation of microbats can be summarized as follows:

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 $\lambda$ 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 $A_{min}$.

In order to apply the bat algorithm for optimization problems more efficiently, some additional assumptions are strongly advisable. In general, we assume that the frequency $f$ evolves on a bounded interval $[f_{min}, f_{max}]$. This means that the wavelength $\lambda$ is also bounded, because $f$ and $\lambda$ are related to each other by the fact that the product $\lambda.f$ is constant. For practical reasons, it is also convenient that the largest wavelength is chosen such that it is comparable to the size of the domain of interest (the search space, for optimization problems). 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. With these idealized rules indicated above, the basic pseudo-code of the bat algorithm is shown in Algorithm 1. It is described in next section.

## 3.2 The Algorithm

Basically, the algorithm considers an initial population of $\mathcal{P}$ individuals (bats). Each bat, representing a potential solution of the optimization problem, has a location $\mathbf{x}_i$ and velocity $\mathbf{v}_i$. The algorithm initializes these variables with random values within the search space. Then, the pulse frequency, pulse rate, and loudness are computed for each individual bat (lines 2-6). Then, the swarm evolves in a discrete way over generations (line 7), like time instances (line 21) until the maximum number of generations, $\mathcal{G}_{max}$, is reached (line 22). For each generation $g$ and each bat (line 8), new frequency, location and velocity are computed (lines 9-10) according to the following evolution equations:

$$f_i^g = f_{min}^g + \beta(f_{max}^g - f_{min}^g) \tag{1}$$

$$\mathbf{v}_i^g = \mathbf{v}_i^{g-1} + [\mathbf{x}_i^g - \mathbf{x}^*]\, f_i^g \tag{2}$$

$$\mathbf{x}_i^g = \mathbf{x}_i^{g-1} + \mathbf{v}_i^g \tag{3}$$

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 $g$.

It is worthwhile to remark a certain similarity of the evolutions equations (2)-(3) to those of the velocity and position in standard particle swarm optimization (PSO), as $f_i$ controls the pace and range of the movement of the particles of the swarm. From this viewpoint, the bat algorithm is a balanced combination of the standard PSO and the local search modulated by the loudness and pulse rate.

The current global best solution and a local solution around it are probabilistically selected according to a given criterion (lines 11-14). Then, search is

```
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 vectors: $\Theta = (\theta_1, \dots, \theta_{\mathcal{P}}), \Psi = (\psi_1, \dots, \psi_{\mathcal{P}})$ with $\theta_k, \psi_k \in U(0,1)$
 1: $g \leftarrow 0$                                                  //$g$: generation index
 2: for $i = 1$ to $\mathcal{P}$ do
 3:    Initialize the location and velocity $\mathbf{x}_i$ and $\mathbf{v}_i$          //Initialization phase
 4:    Define pulse frequency $f_i$ at $\mathbf{x}_i$
 5:    Initialize pulse rates $r_i$ and loudness $\mathcal{A}_i$
 6: end for
 7: while $g \leq \mathcal{G}_{max}$ do
 8:    for $i = 1$ to $\mathcal{P}$ do
 9:       Generate new solutions by adjusting frequency,
10:       and updating locations and velocities                        //eqns. (1)-(3)
11:       if $\theta_i > r_i$ then
12:          $\mathbf{s}^{best} \leftarrow \mathbf{s}^g$                       //select the current best global solution
13:          $\mathbf{ls}^{best} \leftarrow \mathbf{ls}^g$                     //generate a local solution around $\mathbf{s}^{best}$
14:       end if
15:       Generate a new solution by local random walk                    //eqn. (4)
16:       if $\psi_i < \mathcal{A}_i$ and $\phi(\mathbf{x_i}) < \phi(\mathbf{x}^*)$ then
17:          Accept new solutions
18:          Increase $r_i$ and decrease $\mathcal{A}_i$                        //eqns. (5)-(6)
19:       end if
20:    end for
21:    $g \leftarrow g + 1$
22: end while
23: Rank the bats and find current best $\mathbf{x}^*$
24: return $\mathbf{x}^*$
```

**Algorithm 1:** Bat Algorithm

intensified by a local random walk (line 15). For this local search, once a solution is selected among the current best solutions, it is perturbed locally through a random walk of the form:

$$\mathbf{x}_{new} = \mathbf{x}_{old} + \epsilon \mathcal{A}^g \tag{4}$$

where $\epsilon$ is a random number with uniform distribution 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 achieved 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 (lines 16-19). 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. However, any other value within its range can also be feasible (see our discussion about parameter tuning in Section 4.2 for details). The evolution rules for loudness and pulse rate are as follows:

$$\mathcal{A}_i^{g+1} = \alpha \mathcal{A}_i^g \tag{5}$$

$$r_i^{g+1} = r_i^0 [1 - exp(-\gamma g)] \tag{6}$$

where $\alpha$ and $\gamma$ are constants. Note that for any $0 < \alpha < 1$ and any $\gamma > 0$ we have:

$$\mathcal{A}_i^g \to 0, \quad r_i^g \to r_i^0, \quad \text{as } g \to \infty \tag{7}$$

In general, each bat should have different values for loudness and pulse emission rate, which can be computationally achieved by randomization. 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 Proposed Method

### 4.1 Problem To Be Solved

We assume that the reader is familiar with the main concepts of free-form parametric curves [4]. Mathematically, a *free-form rational Bézier curve* $\mathbf{\Phi}(\tau)$ *of degree* $\eta$ is defined as:

$$\mathbf{\Phi}(\tau) = \frac{\displaystyle\sum_{j=0}^{\eta} \omega_j \mathbf{\Lambda}_j \phi_j^\eta(\tau)}{\displaystyle\sum_{j=0}^{\eta} \omega_j \phi_j^\eta(\tau)} \tag{8}$$

where $\mathbf{\Lambda}_j$ are vector coefficients called the *poles*, $\omega_j$ are their scalar weights, $\phi_j^\eta(\tau)$ are the *Bernstein polynomials of index* $j$ *and degree* $\eta$, given by:

$$\phi_j^\eta(\tau) = \binom{\eta}{j} \tau^j \, (1 - \tau)^{\eta - j}$$

and $\tau$ is the *curve parameter*, defined on the finite interval $[0, 1]$. By convention, $0! = 1$. Note that in this paper vectors are denoted in bold.

Suppose now that we are given a set of data points $\{\mathbf{\Delta}_i\}_{i=1,\ldots,\kappa}$ in $\mathbb{R}^\nu$ (usually $\nu = 2$ or $\nu = 3$). Our goal is to obtain the rational Bézier curve $\mathbf{\Phi}(\tau)$ performing discrete approximation of the data points $\{\mathbf{\Delta}_i\}_i$. To do so, we have to compute all parameters (i.e. poles $\mathbf{\Lambda}_j$, weights $\omega_j$, and parameters $\tau_i$ associated with data points $\mathbf{\Delta}_i$, for $i = 1, \dots, \kappa$, $j = 0, \dots, \eta$) of the approximating curve $\mathbf{\Phi}(\tau)$ by minimizing the least-squares error, $\Upsilon$, defined as the sum of squares of the residuals:

$$\Upsilon = \underset{\substack{\{\tau_i\}_i \\ \{\mathbf{\Lambda}_j\}_j \\ \{\omega_j\}_j}}{\text{minimize}} \left[ \sum_{i=1}^{\kappa} \left( \mathbf{\Delta}_i - \frac{\sum_{j=0}^{\eta} \omega_j \mathbf{\Lambda}_j \phi_j^{\eta}(\tau_i)}{\sum_{j=0}^{\eta} \omega_j \phi_j^{\eta}(\tau_i)} \right)^2 \right]. \tag{9}$$

Now, taking:

$$\varphi_j^{\eta}(\tau) = \frac{\omega_j \phi_j^{\eta}(\tau)}{\sum_{k=0}^{\eta} \omega_k \phi_k^{\eta}(\tau)} \tag{10}$$

Eq. (9) becomes:

$$\Upsilon = \underset{\substack{\{\tau_i\}_i \\ \{\mathbf{\Lambda}_j\}_j \\ \{\omega_j\}_j}}{\text{minimize}} \left[ \sum_{i=1}^{\kappa} \left( \mathbf{\Delta}_i - \sum_{j=0}^{\eta} \mathbf{\Lambda}_j \varphi_j^{\eta}(\tau) \right)^2 \right], \tag{11}$$

which can be rewritten in matrix form as:

$$\mathbf{\Omega}.\mathbf{\Lambda} = \mathbf{\Xi} \tag{12}$$

called the *normal equation*, where: $\mathbf{\Omega} = [\Omega_{i,j}] = \left[ \left( \sum_{k=1}^{\kappa} \varphi_i^{\eta}(\tau_k) \varphi_j^{\eta}(\tau_k) \right)_{i,j} \right]$, $\mathbf{\Xi} = [\Xi_j] = \left[ \left( \sum_{k=1}^{\kappa} \mathbf{\Delta}_k \varphi_j^{\eta}(\tau_k) \right)_j \right]$, $\mathbf{\Lambda} = (\mathbf{\Lambda}_0, \dots, \mathbf{\Lambda}_\eta)^T$, for $i, j = 0, \dots, \eta$, and $(.)^T$ means the transposition of a vector or a matrix. In general, $\kappa >> \eta$ meaning that the system (12) is over-determined. If values are assigned to the $\tau_i$, our problem can be solved as a classical linear least-squares minimization, with the coefficients $\{\mathbf{\Lambda}_i\}_{i=0,\dots,\eta}$ as unknowns. This problem can readily be solved by standard numerical techniques. On the contrary, if the values of $\tau_i$ are treated as unknowns, the problem becomes much more difficult. Indeed, since the polynomial blending functions $\phi_j^{\eta}(\tau)$ are nonlinear in $\tau$ and so are the rational blending functions $\varphi_j^{\eta}(\tau)$, the least-squares minimization of the errors is a nonlinear continuous optimization problem. Note also that in many practical cases the number of data points can be extremely large, meaning that we have to deal with a large number of unknowns. In other words, we are also confronted with a high-dimensional problem. It is also a multimodal problem, since there might be arguably more than one set of parameter values leading to the optimal solution.

In conclusion, it is clear that the interplay among all sets of unknowns (data parameters, poles, and weights) leads to a very difficult over-determined, multimodal, multivariate, continuous, nonlinear optimization problem. In this work, we are interested to solve this general problem. This means that we make no assumption about the values of the free parameters; instead, we compute them at full extent.

## 4.2 Proposed Method and Parameter Tuning

Our approach to solve this general problem consists of applying the bat algorithm described above to determine suitable parameter values for the least-squares minimization of functional $\Upsilon$ according to (9). To this aim, each bat, representing a potential solution, corresponds to a parametric vector $\mathcal{S}_j$ of length $\kappa+\eta+1$ given by $\mathcal{S}_j=[\mathcal{T}_j;\mathcal{W}_j]$, where $\mathcal{T}_j=(\tau_1^j,\tau_2^j,\dots,\tau_\kappa^j)\in[0,1]^\kappa$ with the $\{\tau_i^j\}_{i=1,\dots,\kappa}$ strictly increasing parameters, and $\mathcal{W}_j=(w_0^j,\dots,w_\eta^j)$ for $j=1,\dots,\mathcal{P}$. These parametric vectors are initialized with random values. Also, coordinates of $\mathcal{T}_j$ are normalized and then sorted. Regarding $\mathcal{W}_j$, in this paper we consider values for the weights within the range $(0,100)$, as values larger than 100 do not modify the shape of the curve noticeably. Application of the bat algorithm described above yields new positions and velocities of the bats representing the potential solutions of our optimization problem.

A critical issue when working with meta-heuristic techniques is the parameter tuning. It is well-known that the performance of these methods is strongly dependent on the choice of suitable values for their parameters. Moreover, such values are problem-dependent, making it hard to determine good values in advance. Therefore, although there are some papers describing suitable values for some problems, our choice must be necessarily empirical. To this purpose, we carried out numerous computer simulations for different parameter values. The different parameters used in our method are arranged in rows in Table 1. For each parameter, the table shows (in columns) its symbol, meaning, range of values, and the parameter value chosen in this paper.

The most important parameters in the bat algorithm are:

- *population size*: in general, increasing the number of individuals (bats) decreases the number of required iterations, but it also increases the number of function evaluations. Therefore, a trade-off between both situations must be achieved for better performance. In this work, we tested populations ranging from 10 to 200 bats and found that while very low values require too many iterations, values in the range $100-200$ perform similarly, so we set this value to 100 individuals.
- *maximum number of iterations*: we tested our method for values of this parameter in the range $100-3000$ and found that the method converged in less than 1500 iterations in all our executions. We finally set this parameter in 2000 iterations, but to prevent wasting computation time without any improvement, also set an additional termination criterion: the method stops if no further improvement of the solution is reached after 20 consecutive

**Table 1.** Parameters and values used in our method.

| *Symbol* | *Meaning* | *Range of Values* | *Chosen Value* |
|---|---|---|---|
| $\mathcal{P}$ | population size | $10-200$ | 100 |
| $g$ | maximum number of generations | $100-3000$ | 2000 |
| $\mathcal{A}^0$ | initial loudness | $(0,2)$ | 0.5 |
| $\mathcal{A}_{min}$ | minimum loudness | $[0,1]$ | 0 |
| $r^0$ | initial pulse rate | $[0,1]$ | 0.5 |
| $f_{max}$ | maximum frequency | $[0,10]$ | 2 |
| $\alpha$ | multiplicative factor | $(0,1)$ | 0.6 |
| $\gamma$ | exponential factor | $[0,1]$ | 0.4 |

iterations, even although the total number of iterations is less than 2000. With this additional criterion, the computation times improved significantly without penalizing the quality of the final solution.

– *initial and minimum loudness and parameter* $\alpha$: they are set to 0.5, 0, and 0.6, respectively. However, from our computer experiments we noticed that our results do not change significantly for values of the initial loudness in the whole range $(0,2)$, meaning that this parameter is very robust against variations on that interval. Empirically, we found $\alpha = 0.6$ to be a suitable value for this problem.
– initial pulse rate and parameter $\gamma$: of these two parameters, the initial pulse rate is the most relevant. In fact, parameter $\gamma$ only affects the very early iterations. We set the initial pulse rate to 0.5, meaning that the selection has an equal probability of change in the long term.

With this choice of parameters, the bat algorithm is run iteratively for a given number of generations or until the convergence of the minimization of the error is eventually achieved. Final positions and velocities of the bats are computed and ranked according to our fitness function $\Upsilon$. The position of the global best is taken as the final solution of our minimization problem.

## 5 Experimental Results

The method described in previous section has been applied to several examples chosen by the authors. Unfortunately, the field still lacks a standardized benchmark for further analysis and comparative purposes. We think, however, that the examples reported here will be useful to determine the good applicability of our method to this problem. To keep the paper in manageable size, in this section we describe only three of them, corresponding to 2D and 3D curves as described in Table 2. In this table, the three examples are reported in rows. For each example, the table shows (in columns) the number of data points used in our experiments along with some other interesting features: if the curve is open

**Table 2.** Benchmark used in this paper along with the main features of each example.

| | *# data curve* | *Open/closed curve* | *Non-differentiable points* | *2D/3D curve* | *DOFs (degrees of freedom)* |
|---|---|---|---|---|---|
| *Hypocycloid* | 200 | closed | ✓ | 2D | 124 |
| *Spiral* | 100 | open | × | 2D | 136 |
| *Helix* | 100 | open | × | 3D | 242 |

**Table 3.** Fitting errors for the examples used in this paper (arranged in rows): coordinate error, $\Upsilon$ error and $RMSE$ for the mean and best results from 50 executions (in columns).

| *Curve* | $Av_\mu$*(mean)* | $Av_\mu$*(best)* | $\Upsilon$ (mean) | $\Upsilon$ (best) | $RMSE$ (mean) | $RMSE$ (best) |
|---|---|---|---|---|---|---|
| *Hypocycloid* | $x$ : 6.390788$e$-4<br>$y$ : 7.138176$e$-4 | $x$ : 3.294507$e$-5<br>$y$ : 4.051993$e$-5 | 3.475935$e$-4 | 3.075698$e$-5 | 1.318322$e$-3 | 3.921542$e$-4 |
| *Spiral* | $x$ : 6.998090$e$-3<br>$y$ : 7.189277$e$-3 | $x$ : 2.532214$e$-4<br>$y$ : 2.985567$e$-4 | 8.589174$e$-4 | 1.867033$e$-5 | 2.930729$e$-3 | 4.320918$e$-4 |
| *Helix* | $x$ : 1.050018$e$-2<br>$y$ : 1.102567$e$-2<br>$z$ : 9.711053$e$-4 | $x$ : 3.277831$e$-3<br>$y$ : 2.689946$e$-3<br>$y$ : 3.091722$e$-4 | 3.261139$e$-4 | 1.860439$e$-5 | 1.217693$e$-3 | 4.313281$e$-4 |

or closed, planar or 3D, or whether it includes any non-differentiable point (such as cusps or discontinuities). It also reports the number of degrees of freedom (DOFs) of the optimization problem (i.e. the number of variables to be minimized). These examples have been primarily chosen to reflect the diversity of situations our method can be applied to. First example corresponds to a planar closed curve called hypocycloid, which has several cusps; second example shows a spiral, a planar open curve with a smooth yet challenging shape; and last example corresponds to a helix, a 3D open curve with several DOFs.

Figure 1 shows (from top to bottom) the results of our experiments for the three examples in our benchmark. In all cases, the original data points are displayed as red empty circles while the reconstructed points are displayed as blue + symbols. Note the very good matching between both sets of points for the three examples. This good visual behavior is confirmed by our numerical results, reported in Table 3. The three examples are arranged in rows. For each example, the table reports (in columns):

- the average and the best error of the coordinates, according to the equation: $Av_\mu = \sum_{i=1}^{\kappa} |\delta_j^\mu - \bar{\delta_j^\mu}|$, where $\delta_j^\mu$ and $\bar{\delta_j^\mu}$ represent respectively the $\mu$ coordinate of the input data and reconstructed data, with $\mu = x, y$ for 2D curves ($\mu = x, y, z$ for 3D curves);
- the average and best error of the functional $\Upsilon$ according to Eq. (9);

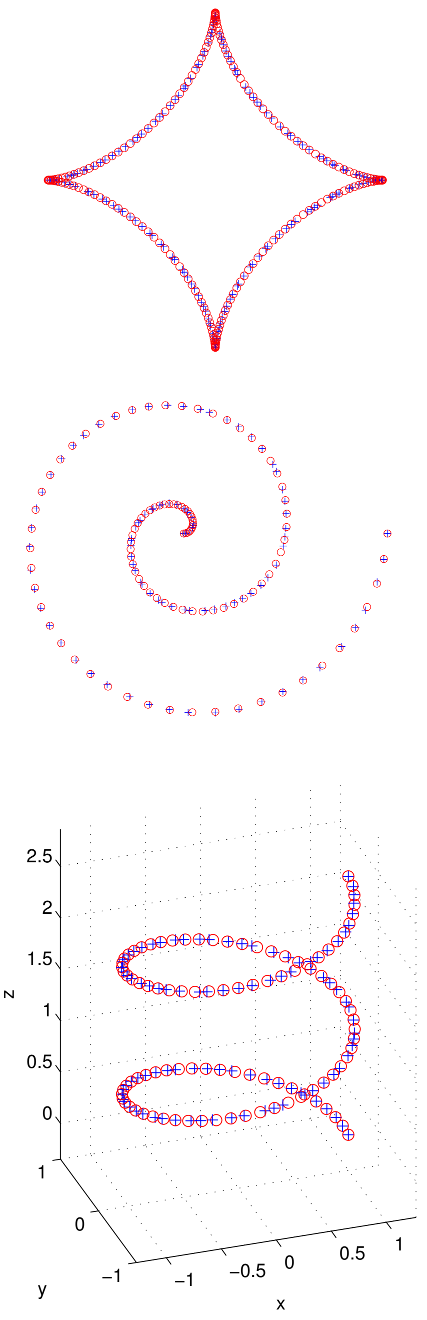

**Fig. 1.** Examples of application of the bat algorithm for data approximation with rational Bézier curves: (top) hypocycloid; (middle) spiral; (bottom) helix. Original points are displayed as red empty circles and the reconstructed points as blue + symbols.

– the average and best error of the root-mean square error, given by:

$$RMSE = \sqrt{\frac{\Upsilon}{\kappa}}.$$

The average error has been obtained as the mean value from 50 independent executions of the algorithm, while the best value corresponds to the results of the best execution from the 50 runs. As the reader can see, average errors are of order $10^{-4}$ (average) to $10^{-5}$ (best) for the $\Upsilon$ error, with RMSE of order $10^{-4}$ (average) to $10^{-5}$ (best) for the three examples in our benchmark. Coordinate errors show more noticeable variations for the third example due to the scale factor of the vertical component. From the data in Table 3 we can conclude that the method performs very well, being able to replicate the original shape with high accuracy for all instances in our benchmark.

## 6 Conclusions and Future Work

This paper introduces a new method for discrete approximation of data points with rational Bézier curves. Given a set of data points, the method computes all relevant parameters (poles, weights, and data parameters) of the rational Bézier fitting curve as the solution of a challenging over-determined nonlinear multimodal multivariate continuous optimization problem. The method is based on a nature-inspired meta-heuristic called bat algorithm, recently introduced to solve difficult optimization problems. To check the performance of our approach, it has been applied to some illustrative examples of 2D and 3D curves. Our results show that the method performs very well, being able to yield a satisfactory approximating curve with a high degree of accuracy. This approach generalizes previous methods for data fitting based on polynomial basis functions to rational blending functions, thus expanding the potential range of applications to include more difficult shapes.

Regarding the implementation issues, all computations in this paper have been performed on a 2.6 GHz. Intel Core i7 processor with 8 GB. of RAM. The source code has been implemented by the authors in the native programming language of the popular scientific program *Matlab*, version 2013b. The systems of equations have been computed by using specialized *Matlab* commands for Gaussian elimination with partial pivoting and singular value decomposition (SVD) for squared and non-squared systems, respectively.

Our future work includes the extension of this method to the case of surfaces. We are also interested to analyze its application to some industrial processes and other interesting real-world problems in different fields. Finally, a theoretical analysis about the convergence of this method and a comparative analysis with other alternative approaches on a standardized benchmark (when available) are also part of our future goals.

**Acknowledgements.** This research has been kindly supported by the Computer Science National Program of the Spanish Ministry of Economy and Competitiveness, Project Ref. #TIN2012-30768, Toho University (Funabashi, Japan), and the University of Cantabria (Santander, Spain). The authors are particularly grateful to the Department of Information Science of Toho University for all the facilities given to carry out this work.

## References

1. Barhak, J., Fischer, A.: Parameterization and reconstruction from 3D scattered points based on neural network and PDE techniques. IEEE Trans. on Visualization and Computer Graphics, 7(1) 1–16 (2001)
2. Barnhill, R.E.: Geometric Processing for Design and Manufacturing. SIAM, Philadelphia (1992)
3. Dierckx, P.: Curve and Surface Fitting with Splines. Oxford University Press, Oxford (1993)
4. Farin, G.: Curves and surfaces for CAGD (5th ed.). Morgan Kaufmann, San Francisco (2002)
5. Echevarría, G., Iglesias, A., Gálvez, A.: Extending neural networks for B-spline surface reconstruction. Lectures Notes in Computer Science, 2330, 305–314 (2002)
6. Gálvez, A., Iglesias A.: Efficient particle swarm optimization approach for data fitting with free knot B-splines. Computer-Aided Design, 43(12) 1683–1692 (2011)
7. Gálvez A., Iglesias A.: Particle swarm optimization for non-uniform rational B-spline surface reconstruction from clouds of 3D data points. Information Sciences, 192(1) 174–192 (2012)
8. Gálvez A., Iglesias A.: A new iterative mutually-coupled hybrid GA-PSO approach for curve fitting in manufacturing. Applied Soft Computing, 13(3) 1491–1504 (2013)
9. Gálvez A., Iglesias A.: Firefly algorithm for polynomial Bézier surface parameterization. Journal of Applied Mathematics, Article ID 237984, 9 pages (2013)
10. Gálvez A., Iglesias A.: From nonlinear optimization to convex optimization through firefly algorithm and indirect approach with applications to CAD/CAM. The Scientific World Journal, Article ID 283919, 10 pages (2013)
11. Gálvez A., Iglesias A.: Firefly algorithm for explicit B-Spline curve fitting to data points. Mathematical Problems in Engineering, Article ID 528215, 12 pages, (2013)
12. Gálvez A., Iglesias A.: Cuckoo search with Lévy flights for weighted Bayesian energy functional optimization in global-support curve data fitting. The Scientific World Journal, Article ID 138760, 11 pages (2014)
13. Gálvez A., Iglesias A.: New memetic self-adaptive firefly algorithm for continuous optimization. International Journal of Bio-Inspired Computation (in press)
14. Gálvez A., Iglesias A., Avila, A.: Immunological-based approach for accurate fitting of 3D noisy data points with Bézier surfaces. In: Proc. of Int. Conference on Comp. Science-ICCS'2013, Procedia Computer Science, 18, 50–59 (2013)
15. Gálvez A., Iglesias A., Avila, A., Otero, C., Arias, R., Manchado, C.: Elitist clonal selection algorithm for optimal choice of free knots in B-spline data fitting. Applied Soft Computing, 26, 90–106 (2015)
16. Gálvez, A., Iglesias A., Puig-Pey J.: Iterative two-step genetic-algorithm method for efficient polynomial B-spline surface reconstruction. Information Sciences, 182(1) 56-76 (2012)
17. Gu, P., Yan, X.: Neural network approach to the reconstruction of free-form surfaces for reverse engineering. Computer-Aided Design, 27(1) 59–64 (1995)

18. Hoffmann M.: Numerical control of Kohonen neural network for scattered data approximation. Numerical Algorithms, 39, 175–186 (2005)
19. Iglesias, A., Echevarría, G., Gálvez, A.: Functional networks for B-spline surface reconstruction. Future Generation Computer Systems, 20(8) 1337–1353 (2004)
20. Iglesias, A., Gálvez, A.: Hybrid functional-neural approach for surface reconstruction. Mathematical Problems in Engineering, Article ID 351648, 13 pages (2014)
21. Jing, L., Sun, L.: Fitting B-spline curves by least squares support vector machines. Proc. of the 2nd. Int. Conf. on Neural Networks & Brain. Beijing (China). IEEE Press, 905–909 (2005)
22. Li, W., Xu, S., Zhao, G., Goh, L.P.: Adaptive knot placement in B-spline curve approximation. Computer-Aided Design, 37, 791–797 (2005)
23. Loucera, C., Gálvez, A., Iglesias, A.: Simulated annealing algorithm for Bézier curve approximation. Proc. of Cyberworlds 2014, IEEE Computer Society Press, Los Alamitos CA, 182-189 (2014)
24. Mirjalili, S., Mirjalili, S.M.,Yang, X.S.: Binary bat algorithm. Neural Computing and Applications, 25, 663–681 (2014)
25. Park, H.: An error-bounded approximate method for representing planar curves in B-splines. Computer Aided Geometric Design, 21, 479–497 (2004)
26. Park, H., Lee, J.H.: B-spline curve fitting based on adaptive curve refinement using dominant points. Computer-Aided Design, 39, 439–451 (2007)
27. Patrikalakis, N.M., Maekawa, T.: Shape Interrogation for Computer Aided Design and Manufacturing. Springer Verlag, Heidelberg (2002)
28. Pottmann, H., Leopoldseder, S. Hofer, M., Steiner, T., Wang, W.: Industrial geometry: recent advances and applications in CAD. Computer-Aided Design, 37, 751–766 (2005)
29. Powell, M.J.D.: Curve fitting by splines in one variable. In: Hayes, J.G. (editor): Numerical approximation to functions and data. Athlone Press, London (1970)
30. Rekaby, A.: Directed artificial bat algorithm (DABA): a new bio-inspired algorithm. Proc. of International Conference on Advances in Computing, Communications and Informatics (ICACCI), Mysore (India), 1241–1246 (2013)
31. Sarfraz, M., Raza, S.A.: Capturing outline of fonts using genetic algorithms and splines. Proc. of Fifth International Conference on Information Visualization IV'2001, IEEE Computer Society Press, 738–743 (2001)
32. Wang, W.P., Pottmann, H., Liu, Y.: Fitting B-spline curves to point clouds by curvature-based squared distance minimization. ACM Transactions on Graphics, 25(2) 214–238 (2006)
33. Yang, X.S.: A new metaheuristic bat-inspired algorithm. In: J. R. Gonzalez et al. (Eds.) Nature Inspired Cooperative Strategies for Optimization (NISCO 2010). Studies in Computational Intelligence, Springer Berlin, 284, Springer, 65-74 (2010).
34. Yang, X. S..: Bat algorithm for multiobjective optimization. Int. J. Bio-Inspired Computation, 3(5), 267-274 (2011).
35. Yang, X.S., Gandomi, A.H.: Bat algorithm: a novel approach for global engineering optimization. Engineering Computations, 29(5), 464–483 (2012).
36. Yang, X.S.: Bat algorithm: literature review and applications. Int. J. Bio-Inspired Computation, 5(3), 141–149 (2013)
37. Yoshimoto F., Harada T., Yoshimoto Y.: Data fitting with a spline using a real-coded algorithm. Computer-Aided Design, 35, 751–760 (2003)
38. Zhao, X., Zhang, C., Yang, B., Li, P.: Adaptive knot adjustment using a GMM-based continuous optimization algorithm in B-spline curve approximation. Computer-Aided Design, 43, 598-604 (2011)