Mansoura Engineering Journal

Volume 48 | Issue 3 Article 6

2023

Swarm Intelligence for Solving Some Nonlinear Differential Equations

Ahmed Elzaghal
Department of Mathematics and Engineering Physics, Faculty of Engineering, Mansoura University, Egypt

Mohammed Mohammed Elgamal
Department of Mathematics and Engineering Physics, Faculty of Engineering, Mansoura University, Egypt

Ahmed H. Eltanboly
Department of Mathematics and Engineering Physics, Faculty of Engineering, Mansoura University, Egypt, eng_ahmed_hazem@mans.edu.eg

Follow this and additional works at: https://mej.researchcommons.org/home

 Part of the Engineering Commons, Ordinary Differential Equations and Applied Dynamics Commons, and the Other Applied Mathematics Commons

Recommended Citation

Elzaghal, Ahmed; Elgamal, Mohammed Mohammed; and Eltanboly, Ahmed H. (2023) "Swarm Intelligence for Solving Some Nonlinear Differential Equations," *Mansoura Engineering Journal*: Vol. 48 : Iss. 3 , Article 6.
Available at: https://doi.org/10.58491/2735-4202.3048

This Original Study is brought to you for free and open access by Mansoura Engineering Journal. It has been accepted for inclusion in Mansoura Engineering Journal by an authorized editor of Mansoura Engineering Journal. For more information, please contact mej@mans.edu.eg.

CASE STUDY

Swarm Intelligence for Solving Some Nonlinear Differential Equations☆

Ahmed Elzaghal [a], Mohammed M. Elgamal [a], Ahmed H. Eltanboly [a,b,*]

[a] Department of Mathematics and Engineering Physics, Faculty of Engineering, Mansoura University, Egypt
[b] Department of Mathematics, Faculty of Basic Science, Galala University, New Galala City, Egypt

Abstract

The Euler method is a well-known numerical technique that is employed for solving initial value problems of ordinary differential equations (DE's). The solution obtained through Euler's method is subject to significant inaccuracies, which tend to propagate with each successive iteration. The Particle Swarm Optimization (PSO) algorithm is a highly effective method for finding optimal solutions to both linear and nonlinear optimization problems. In this particular investigation, the PSO technique was utilized to solve initial value problems associated with ordinary DE's. The Euler method, on the other hand, employs equidistant grid points to approximate solutions, which can result in significant errors and a substantial deviation from the actual solution. The PSO algorithm is a reliable approach for achieving optimal solutions to linear and nonlinear optimization problems, including initial value problems associated with ordinary differential equations. In contrast, the Euler method uses evenly spaced grid points to estimate solutions, which can lead to significant inaccuracies and deviation from the true solution. To address this issue, a new approach was developed using PSO to determine non-uniform grid points that minimize the approximation errors. Increasing the number of non-uniform grid points enhances accuracy and reduces errors. Numerical calculations were performed to compare this approach with traditional Euler formulae, demonstrating its superiority in overcoming existing limitations and delivering numerous benefits.

Keywords: Differential equations (DE), Initial value problems, Swarm optimization (SO)

1. Introduction

It is always a major topic to suggest different approaches for the process of obtaining numerical solutions to the problems concerning the initial and boundary values. This is due to the intricate nature of differential equations (DE's) that arise in science and engineering. Various techniques (Huang and Li, 2010; Kamali et al., 2015; Li et al., 2017; Wu and Xiu, 2019; Yuttanan and Razzaghi, 2019) can be employed to tackle nonlinear differential equations; with one of such technique is being the transformation of these equations into linear ordinary differential equations. Another approach involves directly discretizing the differential equations. Several well-known methods, including finite difference and finite element techniques, have been introduced for addressing this issue (Yuttanan and Razzaghi, 2019; Zienkiewicz and Taylor, 2000; Schober et al., 2019; Edalat, Farjudian, Mohammadian, Pattinson). The two similar methods use the same technique that meshes the domain of the function. Then, by using a feasible approach, the calculation of the approximate solutions at the nodes of the mesh can be accomplished (Cullen and Clarke, 2019; Yang et al., 2019). In this study, we deal

☆ This paragraph of the first footnote will contain the date on which you submitted your paper for review. It will also contain support information, including sponsor and financial support acknowledgment. For example, 'This work was supported in part by the U.S. Department of Commerce under Grant BS123456'.

Received 1 February 2023; revised 27 April 2023; accepted 9 May 2023.
Available online 27 July 2023

* Corresponding author at: Department of Mathematics and Engineering Physics, Faculty of Engineering, Mansoura University, Egypt. Tel.: +201014140449.
E-mail address: Eng_Ahmed_Hazem@mans.edu.eg (A.H. Eltanboly).

https://doi.org/10.58491/2735-4202.3048
2735-4202/© 2023 Faculty of Engineering, Mansoura University. This is an open access article under the CC BY 4.0 license (https://creativecommons.org/licenses/by/4.0/).

with an initial value problem of the first order ordinary differential equation, which is represented as follows:

$$\begin{cases} \dfrac{du(x)}{dx} = f(x,u), x \geq x_0 \\ u(x_0) = u_0 \end{cases} \quad (1)$$

where, u_0 is constant, $f(x,u)$ is given and $u(x)$ is to be solved.

The most popular and simple method to solve this problem is the Euler formula (Kress, 1998) that has at series nodes $(x_0 < x_1 < x_2 < \ldots < x_k < \ldots)$ that can represented in the following form:

$$u_{k+1} = u_k + (x_{k+1} - x_k) * f(x_k, u_k), k = 0, 1, 2, \quad (2)$$

where the term u_k indicates the approximate solution at node x_k that u_k is $u(x_k)$. In Euler method the nodes are always selected to be equidistant according to the formula $x_k = x_0 + kh$ (h is the size step).

It should be mentioned that the Euler method was developed and widely used. One of the disadvantages of that method and its approximate solution is the difference between approximate and actual solution greatly increase with the increase in iteration number k, unless the step size is so small (Kress, 1998; Wen, 2019; Sark and Newton, 2008).

Moreover, in engineering and science problems and in many practical situations, many optimization problems have been created (Sark and Newton, 2008). To find the optimal solution, a lot of methods have been developed such as conjugate direction methods, gradient methods, Newton's method and others (Sark and Newton, 2008; Chong and Zak, 2013). Latterly another methods attracted great attention that called the heuristic approximation methods, such as particle swarm optimization (PSO) which we deal with in this research study (Chong and Zak, 2013; Kennedy and Eberhart, 1995).

PSO algorithm is very effective in solving a lot of nonlinear optimization problems, PSO can be used to solve many engineering problems. The purpose of the present study is to apply the PSO algorithm to find approximate solutions and compare it with the actual ones (Jordehi, 2014; Lin et al., 2019; Zhang and Hui, 2016, 2017). In normal methods it's found that the different distribution of nodes $x_0, x_1, x_2, \ldots, x_k$ leads to different values of u_{k+1}.The purpose of PSO algorithm is to find the optimal distribution of nodes $x_0, x_1, x_2, \ldots, x_k$ to get the most accurate approximate value of u_{k+1} (Kennedy and Eberhart, 1995). To achieve that and get the optimal distribution of nodes, an optimization problem must be created according to the least error estimate. By applying PSO, the optimization problem can be solved and the optimal distribution of nodes can be obtained, which is used to determine the most accurate approximate solutions. Numerical results are carried out to show and compare the approximate solutions of PSO with the actual solutions to show the efficiency of PSO (Jordehi, 2014; Lin et al., 2019; Zhang and Hui, 2016; Kennedy et al., 2001). The paper is organized as follows: Section 2 is dedicated to provide a brief review of the PSO algorithm. In section 3, an optimization problem is being constructed; then handled by two methods that use differential and integration techniques. Section 4, shows the numerical results that are obtained by solving the optimization problem based on the PSO algorithm. A comparing with the actual solution has been also conducted.

2. Brief review on particle swarm optimization (PSO)

Throughout the ages, inspiration has often come from nature, with many things remaining to learn and find out. Between many others, Swarm intelligence draws inspiration from the intelligent collective behavior of bird swarms that are observed in nature. In a flock of birds, each member moves in unison and shares information about food sources, resulting in an optimal hunt for the entire group. By simulating this behavior, we can apply it to find the best solution in a complex problem space. Each 'bird' in the swarm contributes to the search for the optimal solution, and the best solution found by the group is considered the best overall. While this approach is heuristic and cannot guarantee finding the true global optimal solution, it has been shown to be highly effective in finding solutions that are very close to the global optimum (Kennedy and Eberhart, 1995).

In 1995 Kennedy and Eberhard developed the Particle swarm optimization. It is best used to find the minimum or maximum of a function defined on a multidimensional vector space (Kennedy and Eberhart, 1995; Kennedy et al., 2001). Like a flock of birds, we have a flock of particles that moving around in the search space, each particle shares its best solution. The movement of each particle depends on three parameters: its old movement, its best solution (P_{best}), and the best solution of all particles (G_{best}).

$$\vec{V}_{i+1} = \omega \vec{V}_i + c_1 \left(\vec{P}_{best} - \vec{X}_i \right) + c_2 \left(\vec{G}_{best} - \vec{X}_i \right) \quad (3)$$

$$\overrightarrow{X}_{i+1} = \overrightarrow{X}_i + \overrightarrow{V}_{i+1} \quad (4)$$

ω, c_1 and c_2 is constant chosen between 0 and 1.

$\overrightarrow{V}_i$: particle's previous velocity. ($\omega \overrightarrow{V}_i$ is the inertia effect).

$\overrightarrow{P}_{best}$: particle's best position in all previous iterations.

$\overrightarrow{G}_{best}$: best position of all particles in all previous iterations.

Particles' movements is guided by two important parts one is the particles' best known position and second is the entire swarm's particles best known position. When improved positions are found, they eventually guide the movements of the swarm. The method is run as follows: some particles X are chosen randomly in the space of search. Apply in function F(X) (the function we need to find it maximum or minimum value). In the first iteration P_{best} of each particle is the function F at this particle $P_{best\ i}$ equal X_i. G_{best} is X_i at which $F(X_i)$ is maximum if we need to find the maximum value of the function. Then use (3) to calculate the velocity of each particle. Then use (4) to calculate the position of each particle X_{i+1}. If $F(X_{i+1})$ is more than $F(P_{best})$ then new $P_{best} = X_{i+1}$ else P_{best} stay the same. If there is any $F(X_{i+1})$ more than $F(G_{best})$ then X_{i+1} will be the new G_{best} else G_{best} stay the same. Then repeat the previous steps until all particles become equal (Zeng et al., 2016, 2018; Zhang and Li, 2015; Hindmarsh and Petzold, 1995; Zhang et al., 2015; Bonyadi and Michalewicz, 2017; Jain et al., 2018; Gosciniak, 2015; Du and Swamy, 2016; Mustafa Servat Kiran, 2017). At this moment, these particles are the best solution. As shown in Fig. 1.

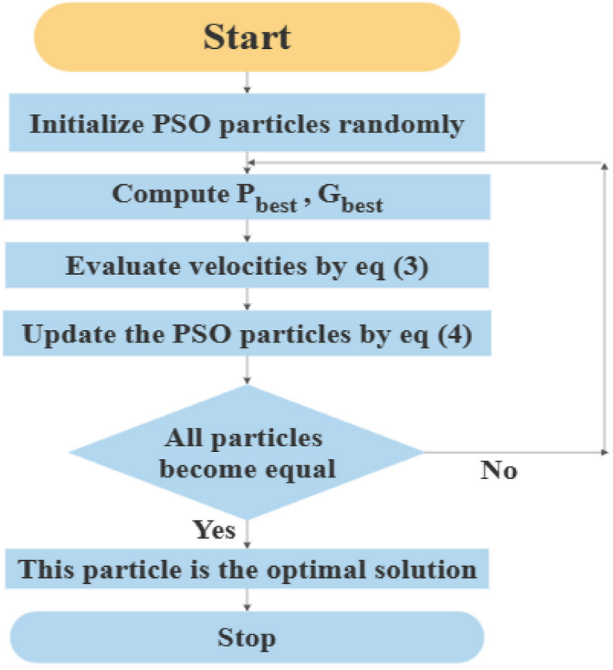

Fig. 1. Flowchart of PSO algorithm.

3. Constructing optimization problem

In this section, we construct many optimization problems to determine the nonequidistant grid points $x_0 < x_1 < x_2 < \ldots < x_n$. Two ways are introduced for solving the initial value problem of ordinary differential equations (Zhong et al., 2019).

1. Differential method. The optimization problem is formed by the differential method. By considering the differential mean value theorem and the interval $[x_k, x_{k+1}]$, there is at least one $\epsilon_k \in [x_k, x_{k+1}]$ such that

$$\frac{u(x_{k+1}) - u(x_k)}{x_{k+1} - x_k} = \frac{du}{dx}\bigg|_{x=\epsilon_k} = f(\epsilon_k, u(\epsilon_k)), k = 0, 1, 2, \quad (5)$$

It is clear that, if the value of ϵ_k is replaced by x_k, equation (5) will be the Euler method. If the value of ϵ_k is replaced by x_{k+1} ,equation (5) will be the implicit Euler method. If the value of ϵ_k can be determined, the application of (5) will lead to a formula as follows:

$$u_{k+1} = u_k + (x_{k+1} - x_k) f(\epsilon_k, u(\epsilon_k)) \quad (6)$$

Unfortunately, the specified value of ϵ_k and $u(\epsilon_k)$ is always difficult to determined, and that is why The Euler method used. Moreover, to construct an optimization problem we have to change the way of thinking as follows:

$$min|u(x_{k+1}) - u_{k+1}| \quad (7)$$

If there is a constant $c_k \in [x_k, x_{k+1}]$ and the unknown ϵ_k is replaced by this constant the optimization problem (7) can be written as:

$$min\left|u(x_k) - u_k + (x_{k+1} - x_k)\left[f(\epsilon_k, u(\epsilon_k)) - f(c_k, u(c_k))\right]\right| \quad (8)$$

Now, we can analyze the optimization problem (8). It is appropriate to assume $u(x_k) = u_k$ meaning that the local truncation error is considered. Defining a new function

$$Q(x_k, x_{k+1}, \epsilon_k, c_k) = \left|(x_{k+1} - x_k)\left[f(\epsilon_k, u(\epsilon_k)) - f(c_k, u(c_k))\right]\right| \quad (9)$$

We have the following result.

Theorem 1. Suppose that $D \subset R^2$ is a domain with f(x, u): $D \longrightarrow R$. If f(x, u) is continuously differentiated on D, the following estimate holds:

$$Q(x_k, x_{k+1}, \epsilon_k, c_k) \le (x_{k+1} - x_k)^2 (M_k + N_k.L_k) \tag{10}$$

where

$$M_k = \max_{x\in[x_k,x_{k+1}]} \left|\frac{\partial f}{\partial x}\right|, L_k = \max_{x\in[x_k,x_{k+1}]} |f|, \quad N_k = \max_{x\in[x_k,x_{k+1}],(x,u)\in D} \left|\frac{\partial f}{\partial u}\right| \tag{11}$$

Proof

according to (10), it follows:

$$\begin{aligned} &|f(\epsilon_k, u(\epsilon_k)) - f(c_k, u(c_k))| = |f(\epsilon_k, u(\epsilon_k)) - f(c_k, u(\epsilon_k)) \\ &+ f(c_k, u(\epsilon_k)) - f(c_k, u(c_k))| = \left|\frac{\partial f(\bar{\epsilon}_k, u(\epsilon_k))}{\partial x}\right| |\epsilon_k - c_k| \\ &+ \left|\frac{\partial f(c_k, \gamma)}{\partial u}\right| |u(\epsilon_k) - u(c_k)| = \left|\frac{\partial f(\bar{\epsilon}_k, u(\epsilon_k))}{\partial u}\right| |\epsilon_k - c_k| \\ &+ \left|\frac{\partial f(c_k, \gamma)}{\partial u}\right| . |f(\gamma_k, u(\gamma_k))| . |\epsilon_k - c_k| \end{aligned} \tag{12}$$

Where $\bar{\epsilon}_k$ and γ_k are located between ϵ_k and c_k; γ lies between $u(\epsilon_k)$ and $u(c_k)$. Then, defining

$$M_k = \max_{x\in[x_k,x_{k+1}]} \left|\frac{\partial f}{\partial x}\right|, L_k = \max_{x\in[x_k,x_{k+1}]} |f|, \quad N_k = \max_{x\in[x_k,x_{k+1}],(x,u)\in D} \left|\frac{\partial f}{\partial u}\right| \tag{13}$$

We have the following relation:

$$|f(\epsilon_k, u(\epsilon_k)) - f(c_k, u(c_k))| \le |\epsilon_k - c_k|(M_k + N_k.L_k) \tag{14}$$

Under the consideration of $\epsilon_k, c_k \in [x_k, x_{k+1}]$, the following result is determined:

$$Q(x_k, x_{k+1}, \epsilon_k, c_k) \le (x_{k+1} - x_k)^2 (M_k + N_k.L_k) \tag{15}$$

The proof is completed.

From theorem 1, It is clear that the values of M_k, N_k and L_K are dependent on the interval $[x_k, x_{k+1}]$.

A new function is built as follows:

$$\overline{Q}_k(x_k, x_{k+1}) = (x_{k+1} - x_k)^2 (M_k + N_k.L_k) \tag{16}$$

In addition, considering the error estimate of the approximate solution u_{k+1}, we give the following result.

Theorem 2. Assume that the function f(x, u) is defined on $D \subset R^2$ with f(x, u): $D \longrightarrow R$. If f(x, u) is continuously differentiated on D, the error estimate is obtained as follows:

$$|u(x_{k+1}) - u_{k+1}| \le \sum_{j=0}^{k} \overline{Q}_j(x_j, x_{j+1}). \tag{17}$$

Proof

$$\begin{aligned} &|u(x_{k+1}) - u_{k+1}| = |u(x_k) - u_k + (x_{k+1} - x_k).[f(\epsilon_k, u(\epsilon_k)) \\ &- f(c_k, u(c_k))]| \le |u(x_k) - u_k| + \overline{Q}_k(x_k, x_{k+1}) \le |u(x_{k-1}) \\ &- u_{k-1}| + \overline{Q}_k(x_k, x_{k+1}) \ldots\ldots\ldots\ldots \le |u(x_0) \\ &- u_0| + \overline{Q}_k(x_k, x_{k+1}) + \overline{Q}_{k-1}(x_{k-1}, x_k) + \ldots + \overline{Q}_0(x_0, x_1) \\ &= \sum_{j=0}^{k} \overline{Q}_j(x_j, x_{j+1}). \end{aligned} \tag{18}$$

The proof is completed.

Applying theorem 2, new optimization problem is constructed as follows:

$$\min_{x_0 < x_1 < \ldots < x_{k+1}} \sum_{j=0}^{k} \overline{Q}_j(x_j, x_{j+1}) \tag{19}$$

When the optimization problem (19) is solved, the nodes x_j (j = 1, 2, …., k+1) can be determined, and there are further used to determine u_j through Euler type method. The approximation solution of $u(x)(x_0 \le x \le x_{k+1})$ is given as

$$u_a(x) = \sum_{j=0}^{k+1} u_j l_j(x) \tag{20}$$

where

$$l_0(x) = \begin{cases} \dfrac{x - x_1}{x_0 - x_1}, x \in [x_0, x_1] \\ 0, x \in [x_1, x_{k+1}] \end{cases}$$

$$l_j(x) = \begin{cases} \dfrac{x - x_{j-1}}{x_j - x_{j-1}}, x \in [x_{j-1}, x_j] \\ \dfrac{x - x_{j+1}}{x_j - x_{j+1}}, x \in [x_j, x_{j+1}] \\ 0, x \notin [x_{j-1}, x_{j+1}] \end{cases} \tag{21}$$

$$l_{k+1}(x) = \begin{cases} \dfrac{x - x_k}{x_{k+1} - x_k}, x \in [x_k, x_{k+1}] \\ 0, x \in [x_0, x_k] \end{cases}$$

The approximate solution u_{k+1} calculated from the previous method is more accurate than the result obtained by the equidistant grid points in Euler method. It means that when we need to calculate approximate solution u_{k+1} with high accuracy, the optimization problem (19) can be solved to determine the distribution of the nonequidistant grid points (Kress, 1998; Wen, 2019; Sark and Newton, 2008; Chong and Zak, 2013; Kennedy and

Eberhart, 1995; Jordehi, 2014; Lin et al., 2019; Zhang and Hui, 2016, 2017; Kennedy et al., 2001; Zeng et al., 2016, 2018; Zhang and Li, 2015).

2. Integration method. Now, to construct an optimization problem using the integration method. We integrate both sides of the differential equation in the initial value problem (1) with respect to x from x_0 to x_{k+1} . The following Volterra integral equation is obtained:

$$u(x_{k+1}) = u_0 + \int_{x_0}^{x_{k+1}} f(x,u)dx. \tag{22}$$

By considering all the grid points $x_0 < x_1 < x_2 < \ldots < x_{k+1}$, integral equation (22) can be rewritten as

$$u(x_{k+1}) = u_0 + \sum_{j=0}^{k} \int_{x_j}^{x_{j+1}} f(x,u)dx \tag{23}$$

Under the assumption of a continuous function $f(x,u)$ on a domain D, there is at least one ϵ_j which $\epsilon_j \in [x_j . x_{j+1}]$ such that

$$u(x_{k+1}) = u_0 + \sum_{j=0}^{k} (x_{j+1} - x_j) f(\epsilon_j, u(\epsilon_j)), \tag{24}$$

where the integral mean value theorem has been applied (23). When the values of ϵ_j and $u(\epsilon_j)$ can be given explicitly, formula (24) is also explicit. However, it is always difficult to give the explicit values of ϵ_j and $u(\epsilon_j)$. Then it is assumed that ϵ_j and $u(\epsilon_j)$ are replaced by c_j and $u(c_j)$, respectively. Hence, the optimization problem is constructed as follows:

$$\begin{aligned}
&min|u(x_{k+1} - u_{k+1}| = \min_{c_j \in [x_j, x_{j+1}]} \left| \sum_{j=0}^{k} (x_{j+1} - x_j) [f(\epsilon_j, u(\epsilon_j)) \right. \\
&\left. - f(c_j, u(c_j)) \right| = \min_{c_j \in [x_j, x_{j+1}]} \left| \sum_{j=0}^{k} \int_{x_j}^{x_{j+1}} f(x,u)dx \right. \\
&\left. - \sum_{j=0}^{k} (x_{j+1} - x_j) f(c_j, u(c_j)) \right| = \min_{c_j \in [x_j, x_{j+1}]} \left| \sum_{j=0}^{k} \int_{x_j}^{x_{j+1}} f(x,u)dx \right. \\
&\left. - \sum_{j=0}^{k} \int_{x_j}^{x_{j+1}} f(c_j, u(c_j))dx \right| = \min_{c_j \in [x_j, x_{j+1}]} \left| \sum_{j=0}^{k} \int_{x_j}^{x_{j+1}} [f(x,u) \right. \\
&\left. - f(c_j, u(c_j))] dx \right|
\end{aligned} \tag{25}$$

Moreover, we introduce the Lipschitz conditions as follows:

$$\begin{aligned}
&|f(x,u) - f(c_j,u)| \le \overline{L}_j |x - c_j|, x, c_j \in [x_j, x_{j+1}], \\
&\forall u |f(x, u(x)) - f(x, u(c_j))| \le \tilde{L}_j |u(x) - u(c_j)|, \\
&\forall x \in [x_j, x_{j+1}]
\end{aligned} \tag{26}$$

Where $\overline{L}_j$ and $\tilde{L}_j$ are positive Lipschitz constants. It gives

$$\begin{aligned}
&\left| \sum_{j=0}^{k} \int_{x_j}^{x_{j+1}} [f(x,u) - f(c_j, u(c_j))] dx \right| = \\
&\left| \sum_{j=0}^{k} \int_{x_j}^{x_{j+1}} [f(x,u) - f(c_j, u(x)) + f(c_j, u(x)) \right. \\
&\left. - f(c_j, u(c_j))] dx \right| \le \left| \sum_{j=0}^{k} \int_{x_j}^{x_{j+1}} (|f(x,u) - f(c_j, u(x))| \right. \\
&\left. + |f(c_j, u(x)) - f(c_j, u(c_j))|) dx \right| = \sum_{j=0}^{k} \int_{x_j}^{x_{j+1}} (\overline{L}_j |x - c_j| \\
&+ \tilde{L}_j |u(x) - u(c_j)|) dx
\end{aligned} \tag{27}$$

In addition, applying the differential mean value theorem, there is at least one γ_j such that

$$|u(x) - u(c_j)| = \left| f\left(\gamma_j, u\left(\gamma_j\right)\right) \right| . |x - c_j| \tag{28}$$

Where γ_j is located between x and c_j. By considering

$$L_j = \max_{x \in [x_j, x_{j+1}]} |f|, \tag{29}$$

we have

$$\begin{aligned}
&\sum_{j=0}^{k} \int_{x_j}^{x_{j+1}} (\overline{L}_j |x - c_j| + \tilde{L}_j |u(x) - u(c_j)|) \, dx \\
&= \sum_{j=0}^{k} \int_{x_j}^{x_{j+1}} \left(\overline{L}_j |x - c_j| + \tilde{L}_j \left| f\left(\gamma_j, u\left(\gamma_j\right)\right) \right| . |x - c_j| \right) dx \\
&\le \sum_{j=0}^{k} \int_{x_j}^{x_{j+1}} (\overline{L}_j + L_j . \tilde{L}_j) |x - c_j| \, dx
\end{aligned} \tag{30}$$

Applying the Cauchy-Schwarz inequality to (28) together with (27), the following relation is given:

$$\left|\sum_{j=0}^{k}\int_{x_j}^{x_{j+1}}\left[f(x,u)-f\left(c_j,u(c_j)\right)\right]dx\right|$$
$$\leq\sum_{j=0}^{k}\left(\overline{L}_j+L_j.\tilde{L}_j\right)\int_{x_j}^{x_{j+1}}|x-c_j|dx$$
$$\leq\sum_{j=0}^{k}\left(\overline{L}_j+L_j.\tilde{L}_j\right)\left[\int_{x_j}^{x_{j+1}}1^2dx.\int_{x_j}^{x_{j+1}}|x-c_j|^2dx\right]^{\frac{1}{2}}$$
$$=\sum_{j=0}^{k}\left(\overline{L}_j+L_j.\tilde{L}_j\right)\left\{\frac{x_{j+1}-x_j}{3}\left[\left(x_{j+1}-c_j\right)^3+\left(x_j-c_j\right)^3\right]\right\}^{1/2} \tag{31}$$

Then, the optimization problem (25) is transformed to the following form:

$$\min_{c_j\in[x_j,x_{j+1}]}\sum_{j=0}^{k}\left(\overline{L}_j+L_j.\tilde{L}_j\right)\left\{\frac{x_{j+1}-x_j}{3}\left[\left(x_{j+1}-c_j\right)^3+\left(x_j-c_j\right)^3\right]\right\}^{1/2} \tag{32}$$

It is easy to get the optimal solution of (32) as

$$\varphi=\frac{\sqrt{3}}{6}\sum_{j=0}^{k}\left(\overline{L}_j+L_j.\tilde{L}_j\right)\left(x_{j+1}-x_j\right)^2 \tag{33}$$

with $c_j=(x_j+x_{j+1})/2$ this means that when the optimization problem (32) is solved, the approximate solution of $u(x_{k+1})$ can be determined as follows:

$$u_{k+1}=u_0+\sum_{j=0}^{k}\left(x_{j+1}-x_j\right)f\left(\frac{x_j+x_{j+1}}{2},u\left(\frac{x_j+x_{j+1}}{2}\right)\right) \tag{34}$$

To put it differently, the midpoint formula (34) was obtained through the solution of an optimization problem. It is worth noting that in this case, the arrangement of grid points is not constrained, making it a crucial aspect. As a result, a fresh optimization problem is formulated as follows:

$$F=\min_{x_0<x_1<\ldots<x_{k+1}}\varphi \tag{35}$$

In other words, the optimization problem (35) determines the distribution of grid points, enabling the use of formula (34) and the Euler method to approximate the solution u_{k+1}. This approach offers the potential for greater accuracy than using equidistant grid points. However, formula (34) includes an unknown term, $u(x_j + x_{j+1}/2$.Which can be simplified using the Euler formula.

$$u\left(\frac{x_j+x_{j+1}}{2}\right)\approx u_k+\frac{x_{j+1}-x_j}{2}f(x_k,u_k) \tag{36}$$

Formula (34) is rewritten as

$$u_{k+1}=u_0+\sum_{j=0}^{k}\left(x_{j+1}-x_j\right).f\left(\frac{x_j+x_{j+1}}{2},\frac{x_{j+1}-x_j}{2}f\left(x_j,u_j\right)\right) \tag{37}$$

As compared with the differential method, the obtained functions (19) and (33) are similar, when the Lipschitz constants $\overline{L}_j$ and $\tilde{L}_j$ are replaced by M_k and N_k, respectively.

As shown, the two optimization problems (19) and (35) are constructed to determine the nonequivalent girds $x_0<x_1<\ldots<x_k$ such that the approximate solution u_{k+1} to $u(x_{k+1})$ has more accuracy than that by using the equivalent grids. Furthermore, when we consider the grids $a=x_0<x_1<\ldots<x_n=b$ and the error estimate on all the grids x_k $(k=1,2\ldots,n)$, the optimization problem (7) can be reread as

$$min\sum_{k=0}^{n-1}|u(x_{k+1})-u_{k+1}| \tag{38}$$

4. Numerical results

This section presents numerical results to demonstrate the efficacy of the PSO method.

Example: Initial value problem of the first order ordinary differential equation. Consider the following problem:

Example 1:

$$\begin{cases}\dfrac{du(x)}{dx}=2x^3-\dfrac{u}{x},x>1\\ u(1)=\dfrac{4}{5}\end{cases} \tag{39}$$

With the exact solution

$$u(x)=\frac{2}{5}x^4+\frac{2}{5x} \tag{40}$$

First, let us construct an optimization problem according to (19) in terms of (39)

$$f(x,u)=2x^3-\frac{u}{x} \tag{41}$$

Meaning that

$$\left|\frac{\partial f}{\partial x}\right|=\left|6x^2+\frac{u}{x^2}\right|\left|\frac{\partial f}{\partial u}\right|=\frac{1}{|x|} \tag{42}$$

We further have the following estimates:

$$M_k=\max_{x\in[x_k,x_{k+1}]}\left[6x^2+\frac{u}{x^2}\right]\leq 6x_{k+1}^2+\frac{1}{x_k^2}\cdot\max_{x\in[x_k,x_{k+1}]}|u|$$

$$N_k = \max_{x\in[x_k,x_{k+1}]} \frac{1}{|x|} = \frac{1}{x_k}$$

$$L_k = \max_{x\in[x_k,x_{k+1}]} \left|2x^3 - \frac{u}{x}\right| \le 2x_{k+1}^3 + \frac{1}{x_k} \max_{x\in[x_k,x_{k+1}]} |u| \tag{43}$$

Then, it follows

$$M_k + N_k.L_k \le 6x_{k+1}^2 + \frac{2x_{k+1}^3}{x_k} + \frac{2}{x_k^2} \max_{x\in[x_k,x_{k+1}]} |u| \tag{44}$$

From the last relation the value of $\max_{x\in[x_k,x_{k+1}]} |u|$ is unknown. Hence, the main term is related to the nodes, which extracted to yield the following function:

$$F = \sum_{j=0}^{k} (x_{j+1} - x_j)^2 \left(6x_{j+1}^2 + \frac{2x_{j+1}^3}{x_j} + \frac{2}{x_j^2}\right) \tag{45}$$

The optimization problem

$$\min_{x_0<x_1<\ldots<x_{k+1}} F \tag{46}$$

is used to determine the values of the nodes. The computation formula to determine the value of u(x) as follows:

$$u_{k+1} = u_0 + \sum_{j=0}^{k} (x_{j+1} - x_j) \left[2\left(\frac{x_j + x_{j+1}}{2}\right)^3 - \frac{2}{x_{j+1} + x_j}\left(\frac{3x_j - x_{j+1}}{2x_j} u_j + (x_{j+1} - x_j)x_j^3\right)\right] \tag{47}$$

By using Particle Swarm Optimization, we can find optimal solution of (46). For numerical computations it is assumed that $\omega = 0.2$ and the constant $c_1 = c_2 = 0.5$. The particle x is considered as (k+2) dimensional vector:

$$\vec{x} = (x_0, x_1, \ldots, x_{k+1}) \tag{48}$$

Subject to the condition

$$1 = x_0 < x_1 < \ldots < x_k < x_{k+1} = c \tag{49}$$

Where c is a specified constant. The number of the swarm's particles is 20. Assumed k = 4 and $x_{k+1} = 2$. We can see from Fig. 2 that there is a rapid decreasing in the values of F then tends to a constant value with the increasing in the generation number. When 'F' become a constant value that means an optimal and stable solution has been obtained. The nonequidistant grid points that have been computed are given as follows:

$$(1, 1.23717507, 1.47559418, 1.68708177, 1.8446576, 2) \tag{50}$$

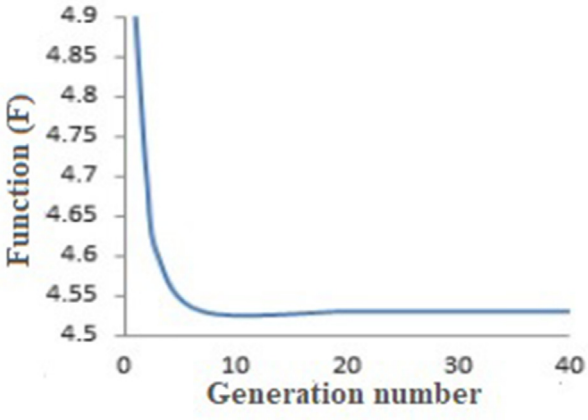

Fig. 2. The relation between F and generation number.

By using (47) and the nonequidistant grid points, the approximate solution can be obtained as $u_5 = 6.609287$. When the equidistant grid points and Euler method we obtain $\tilde{u}_5 = 5.6813$. When the equidistant grid points and Runge-Kutta Method were used we obtained $\overline{u}_5 = 6.5998$.The exact solution is $u(2) = 6.6$. Clearly, the solution $u_5 = 6.609287$ which determined by PSO is more approximate to the exact solution $u(2) = 6.6$ than $\tilde{u}_5 = 5.6813$ which is determined by Euler Method but slightly less approximate than $\overline{u}_5 = 6.5998$ which determined by Runge-Kutta. Accordingly, we report that the PSO method more accurate than the Euler method. However, in linear differential equations, the Runge-Kutta method is slightly more accurate than the PSO method.

To be sure that PSO has really low errors, the values of u_5 computed at $x_{k+1} = 2, 3, 4, 5, 6$ with $k = 4$ and shown in Table 1. It is found from Table 1 that the values of u_5 computed by PSO is really close to the exact ones $u(x_{k+1})$ and the error percentage doesn't exceed 0.25% even if the number of nonequidistant grid points still only 6 as shown in Fig. 3.

Let's discover the effect of increasing the number of nonequidistant grid points on the error and error percentage. Let's determine the approximate value of u_5 at $x_{k+1} = 5$ using $6, 7, 8, 9, 10$ nonequidistant

Table 1. Comparisons between PSO results and exact solution at different x_{k+1}.

x_{k+1}	Exact solution	u_5	Error	Error percentage	Consumed Time (s)
2	6.6	6.6092	0.0092	0.139%	7.96e-5
3	32.533	32.6008	0.0478	0.147%	4.57e-5
4	102.5	102.7044	0.2044	0.199%	4.53e-5
5	250.08	250.6249	0.5449	0.218%	4.67e-5
6	518.4667	519.4615	0.9948	0.192%	5.19e-5

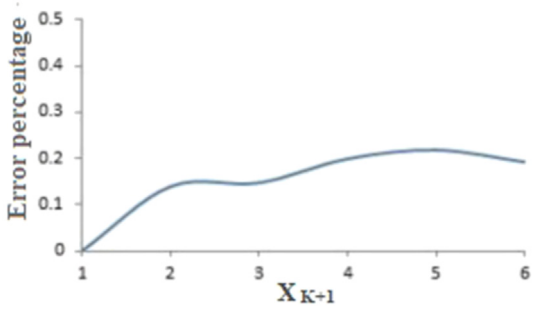

Fig. 3. The distribution of Error percentage versus X_{K+1}.

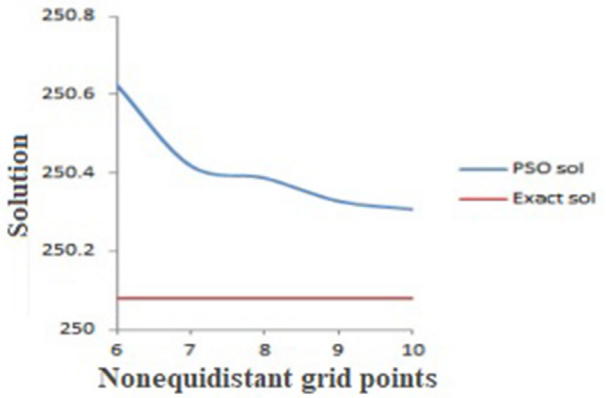

Fig. 4. The variation of values of approximate solution and the exact solution versus the number of nonequidistant grid points.

Table 2. Comparisons between PSO results and Euler results and their errors at different x_{k+1}.

x_{k+1}	Exact solution	PSO	Error of PSO	Euler	Error of Euler
2	6.6	6.6092	0.0092	6.60918	0.00918
3	32.533	32.6008	0.0478	32.6043	0.0713
4	102.5	102.7044	0.2044	102.7571	0.2571
5	250.08	250.6249	0.5449	250.7494	0.6694
6	518.4667	519.4615	0.9948	519.9069	0.4402

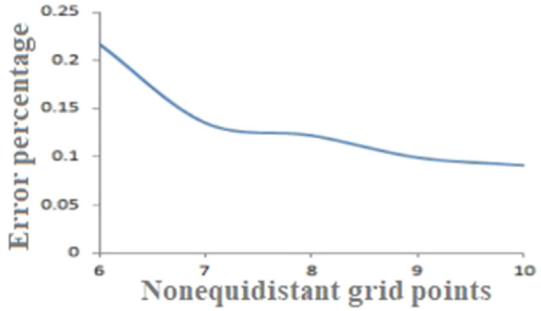

Fig. 5. The variation of error percentage versus the number of nonequidistant grid points.

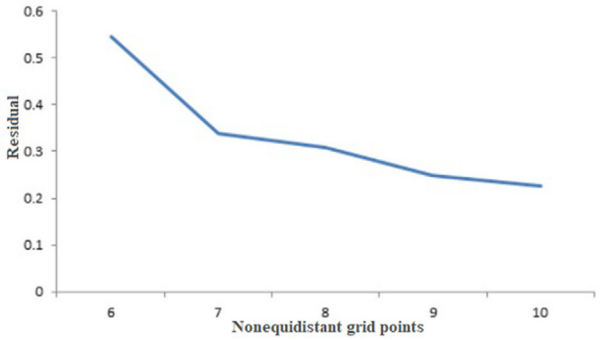

Fig. 6. The residual versus the number of nonequidistant grid points.

Table 3. Comparisons between the approximate values of u_5 at different number of grid points.

Number of nonequidistant grid points	Exact solution	u_5	Error	Error percentage	Consumed Time (s)
6	250.08	250.6249	0.5449	0.217%	4.67e-5
7	250.08	250.4188	0.3388	0.135%	5e-5
8	250.08	250.3873	0.3073	0.122%	6.12e-5
9	250.08	250.3282	0.2482	0.099%	7.77e-5
10	250.08	250.3074	0.2274	0.091%	8.27e-5

grid points. We can see from Fig. 4. That increasing the number of nonequidistant grid points increases the accuracy of approximate solution. In Table 2 we can see the effect of increasing the number of nonequidistant grid points on the accuracy of the approximate solution obtained by PSO. It is clear that increasing the number of nonequidistant grid points increase the accuracy of the obtained values of $u(x_{k+1})$. The error percentage decreases with the increase on nonequidistant grid point as shown in Fig. 5. Now, let's see the effect of increasing the number of nonequidistant grid points on the residual as shown in Fig. 6, Table 3.

5. Conclusion

In Euler method always deals with equidistant grid points for numerically solving ordinary differential equations and that make the approximate solution far from the exact one and with the increasing of iteration number, the error of approximate solution versus the exact one increase, the approximate solution could be greatly away from the exact solution. In PSO method deals with nonequidistant grid points which make the approximate solution really close to the exact one with a low error percentage. The main findings are given as follows.

(1) Using nonequidistant grid point can increase the accuracy of the approximate solution and be more effective that equidistant grid points of Euler method.
(2) The nonequidistant grid points' positions can be determined by constructing and solving an optimization problem depending on the error estimate.
(3) Increasing the number of nonequidistant grid points increases the accuracy of the approximate solution and decreases the error percentage.
(4) Runge-Kutta method has a great accuracy in determining the approximate solution and slightly better than the nonequidistant grid points of PSO in solving linear first order ordinary differential equation.

In conclusion, the study proves that using nonequidistant grid points increasing the accuracy of the approximate solution. Increasing the number of nonequidistant grid points decreases the error. PSO method more accurate than Euler method.

In the future work, the PSO method could be used to develop some methods which using in solving ordinary and partial differential equation like finite element method. Making a comparison between Runge-Kutta and PSO methods in solving nonlinear ordinary differential equations and comparing their accuracies.

The list of equations legend.

1	General form of ODE
2	General Eular formula to solve ODE
3	The velocity of the new particle
4	The new position of the new particle
5–21	Constructing optimization problem using differential method
5	the differential mean value theorem
7	The general form of optimization problem
22–38	Constructing optimization problem using integration method
26	Lipschitz conditions
31	Cauchy-Schwarz inequality
33	Optimal solution
38	Error estimation
39–49	Example and its solution
39	Solved example
40	Exact solution of the solved example
41–46	Construct optimization problem
47	The computation formula to determine the value of u(x)
50	Nonequidistant grid points

Authors contribution

Ahmed Elzaghal: He contributed to Conception and design of the work, data collection and tools, methodology, drafting of the article, data analysis and funding.

Ahmed Eltanboly: He contributed to the methodology, supervision, guide throw paper writing, critical revision of the article and final approval of the version to be published.

Mohamed El-Gamel: He contributed to the methodology, academic supervision and critical revision of the article.

Conflict of interest

The author declared that there are no potential conflicts of interest with respect to the research authorship or publication of this article.

References

Bonyadi, M.R., Michalewicz, Z., 2017. Impacts of coefficients on movement patterns in the particle swarm optimization algorithm. IEEE Trans. Evol. Comput. 21, 378–390.
Chong, E.K.P., Zak, S.H., 2013. An Introduction to Optimization, fourth ed. John Wiley & Sons, Hoboken, NJ, USA.
Cullen, A.C., Clarke, S.R., 2019. A fast, spectrally accurate homotopy based numerical method for solving nonlinear differential equations. J. Comput. Phys. 385, 106–118.
Du, K.L., Swamy, M.N.S., 2016. Particle swarm optimization, pp. 153–173.
Edalat A, Farjudian A, Mohammadian M, Pattinson D. Domain Theoretic Second-Order Euler's method for solving initial value problems. Electron. Notes Theor. Comput. Sci.; 352: 105–128.
Gosciniak, I., 2015. A new approach to particle swarm optimization algorithm. Expert Syst. Appl. 42, 844–854.
Hindmarsh, A.C., Petzold, L.R., 1995. Algorithms and software for ordinary differential equations and differential-algebraic equations, Part I: Euler methods and error estimation. Comput. Phys. 9, 34–41.
Huang, Y., Li, X.-F., 2010. Approximate solution of a class of linear integro-differential equations by Taylor expansion method. Int. J. Comput. Math. 87, 1277–1288.
Jain, N.K., Nangia, U., Jian, J., 2018. A review on particle swarm optimization. J. Inst. Eng. B 99, 407–411.
Jordehi, A.R., 2014. Particle swarm optimisation for dynamic optimisation problems: a review. Neural Comput. Appl. 25, 1507–1516.
Kamali, M.Z.M., Kumaresan, N., Ratnavelu, K., 2015. Solving differential equations with ant colony programming. Appl. Math. Model. 39, 3150–3163.
Kennedy, J., Eberhart, R., 1995. Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks, vol. 4. IEEE Press, Perth, Australia, pp. 1942–1948.
Kennedy, J., Eberhart, R.C., Shi, Y., 2001. Swarm Intelligence. Morgan Kaufmann Publishers, Burlington, MA, USA.
Kiran, Mustafa Servet, 2017. Particle swarm optimization with a new update mechanism, vol. 60, pp. 670–678.
Kress, R., 1998. Numerical Analysis. Springer-Verlag, Berlin, Germany.
Li, X., Li, H., Wu, B., 2017. A new numerical method for variable order fractional functional differential equations. Appl. Math. Lett. 68, 80–86.
Lin, A., Sun, W., Yu, H., Wu, G., Tang, H., 2019. Global genetic learning particle swarm optimization with diversity enhancement by ring topology. Swarm Evol. Comput. 44, 571–583.
Sark, R.A., Newton, C.S., 2008. Optimization Modelling: A Practical Approach. Taylor & Francis Group, New York, NY, USA.

Schober, M., Sarkka, S., Hennig, P., 2019. A probabilistic model for the numerical solution of intitial value problems. Stat. Comput. 29, 99–122.
Wen, H., 2019. Convergence rates of full-implicit truncated Euler–Maruyama method for stochastic differential equations. J Appl Math Comput 60, 147–168.
Wu, K., Xiu, D., 2019. Numerical aspects for approximating governing equations using data. J. Comput. Phys. 384, 200–221.
Yang, C., Deluzet, F., Narski, J., 2019. On the numerical resolution of anisotropic equations with high order differential operators arising in plasma physics. J. Comput. Phys. 386, 502–523.
Yuttanan, B., Razzaghi, M., 2019. Legendre wavelets approach for numerical solutions of distributed order fractional differential equations. Appl. Math. Model. 70, 350–364.
Zeng, N., Wang, Z., Zhang, H., Alsaadi, F.E., 2016. A novel switching delayed PSO algorithm for estimating unknown parameters of lateral flow immunoassay. Cogn Comput 8, 143–152.
Zeng, N., Qiu, H., Wang, Z., Liu, W., Zhang, H., Li, Y., 2018. A new switching-delayed-PSO-based optimized SVM algorithm for diagnosis of Alzheimer's disease. Neurocomputing 320, 195–202.
Zhang, H., Hui, Q., 2016. Parallel multiagent coordination optimization algorithm: implementation, evaluation, and applications. IEEE Trans. Autom. Sci. Eng. 14, 984–995.
Zhang, H., Hui, Q., 2017. Cooperative bat searching algorithm: a combined perspective from multiagent coordination and swarm intelligence. In: Proceedings of the 2017 13th IEEE Conference on Automation Science and Engineering (CASE). School of Civil Engineering and Architecture, Guangxi University, Nanning, Guangxi, China.
Zhang, P.W., Li, T.J., 2015. Numerical analysis. In: Chinese. Peking University Press, Beijing, China.
Zhang, Y., Wang, S., Ji, G., 2015. A comprehensive survey on particle swarm optimization algorithm and its applications, published in Hindawi. Math. Probl Eng. 2015, 1–38.
Zhong, X.-C., Chen, J.-Y., Fan, Z.-Y., 2019. A Particle Swarm Optimization-Based Method for Numerically Solving Ordinary Differential Equations.
Zienkiewicz, O.C., Taylor, R.L., 2000. 1e Finite Element Method, fifth ed. Butterworth-Heinemann, Oxford, UK.