Evolutionary Intelligence (2019) 12:677–688
https://doi.org/10.1007/s12065-019-00275-w

RESEARCH PAPER

# A direct method for solving calculus of variations problems using the whale optimization algorithm

**Seyed Hamed Hashemi Mehne**[1] · **Seyedali Mirjalili**[2]

Received: 26 November 2018 / Revised: 25 July 2019 / Accepted: 6 August 2019 / Published online: 19 August 2019


**Abstract**
A numerical algorithm for solving problems of calculus of variations is proposed and analyzed in the present paper. The method is based on direct minimizing the functional in its discrete form with finite dimension. To solve the resulting optimization problem , the recently proposed whale optimization algorithms is used and adopted. The method proposed in this work is capable of solving constrained and unconstrained problems with fixed or free endpoint conditions. Numerical examples are given to check the validity and accuracy of the proposed method in practice. The results show the superior accuracy and efficiency of the proposed technique as compared to other numerical methods.

**Keywords** Calculus of variations · Whale optimization algorithm · Numerical solution · Finite difference

## 1 Introduction

Calculus of variations deals with problems of optimizing functionals over a given class of functions. As many performance indices can be presented by functionals, there appear widespread formulations as calculus of variations models. Some of practical engineering applications are namely in the fields of image recovery [3], electromagnetics [32], management and economics [10, 17], aerospace [35], and chemical engineering [7].

Alongside with growing of applications, numerical methods for solving variational problems have been also developed. The main motivation in driving numerical methods is that solving these problems analytically is complex and even impossible.

Generally speaking, the methods for solving calculus of variation problems are categorized as direct and indirect methods. These two classes and their methods are discussed in the following paragraphs:

*Direct methods:* In a numerical method if the original problem is converted to a finite-dimensional optimization problem the method is called direct. The approximate optimum is found by an optimization algorithm. The earliest method of this type is the Rayleigh–Ritz method that is based on approximating the solution as a linear combination of unknown functions [12]. The method simplifies the problem to a finite-dimensional optimization one. The Galerkin method is also another similar direct method in which the residual of the Euler–Lagrange is minimized [12]. A direct method of solving calculus of variations problems was also presented in [13]. It is based on a discretization of the Euler–Lagrange equations and using Newton iterative method. In [29], Legendre wavelets are used as base functions for parametrization of the solution. The unknown coefficients in the approximate solution are calculated using necessary conditions for the minimum. In [14] a direct method is proposed based on a non-classical parametrization of the solution as a combination of orthogonal polynomials with unknown coefficients as problem parameters. This approach leads to an optimization problem, and its solution is obtained from a set of algebraic equations. For more attempts to develop direct methods for variational problems, one may refer to [15, 28].

*Indirect methods:* A numerical method based on necessary conditions for the extreme solution that does not directly deal with minimizing functionals is an indirect method. Most of the indirect methods try to solve the

✉ Seyed Hamed Hashemi Mehne
hmehne@ari.ac.ir

[1] Aerospace Research Institute, Tehran 14665-834, Iran

[2] Institute for Integrated and Intelligent Systems, Griffith University, Nathan Campus, Brisbane, QLD 4111, Australia

well-known Euler–Lagrange equations with boundary conditions. For example, the Chebyshev finite difference method is applied to the boundary value problem arising from the Euler–Lagrange equation in [31] to find a numerical solution. In [26], the differential transform method (DTM) is used to solve the corresponding Euler–Lagrange equation. The unknown coefficients, constructing the solution, are found in a recursive manner. Being a method for solving boundary value problems, homotopy perturbation is used in [2] for approximating the solution of variational problems. Similar approach by Adomian decomposition was presented in [5]. In [23] the interpolation based on the exponential spline was used to reduce the Euler–Lagrange equation to a system of algebraic equations. The solution of this system determines the parameters in the approximate solution. Other notable indirect methods were reported in [6, 36, 37]

The main drawback of indirect methods is that they find usually extremum solutions which may not be optimum. They also restricted to special classes of differentiable functions.

In the present paper, a direct method for solving calculus of variations problems with ordinary derivatives is proposed. The original infinite-dimensional problem is first converted to a finite-dimensional optimization problem based on time discretization. Then, the resulting problem is solved by using the whale optimization algorithm (WOA) [21] which is a new population-based meta-heuristic. This algorithm has been widely used in different fields after publication and tends to outperform a large number of similar algorithms in the literature on different optimization problems. What makes this algorithm different from others is the unique search mechanism inspired by the bubble-net hunting behavior of humpback whales, simplicity of the algorithm, and a small number of tuning parameters. In fact, this algorithm has been equipped with several adaptive, time-varying parameters to change its search pattern depends on the shape of a search space for a given optimization problem. This motivated our attempts to investigate the merits of this algorithm for solving problems of the calculus of variations and develop a numerical algorithm in this area using this algorithm.

The proposed method will be evaluated in different case studies. Results show that the method is reliable and accurate. The method has some benefits in comparison with the above-mentioned methods:

- It does not restrict to a special class of functions and leads to alternative solutions
- The proposed method adopts with free endpoint conditions
- Constrained problems can be solved as well

The rest of the paper is organized as follows:

Section 2 presents and formulates the problem and the way it is discretized. The proposed technique is discussed in Sect. 3. Section 4 provides results and relevant discussions. Finally, the conclusions and future works are given in Sect. 5.

## 2 Problem statement and discretization

Let consider the general form of a calculus of variations problem including of minimizing a functional such as follows:

$$J(x,\dot{x}) = \int_0^{t_f} f(t,x(t),\dot{x}(t))dt \tag{1}$$

It has usually fixed initial and final conditions like:

$$x(0) = x_0,\ x(t_f) = x_f, \tag{2}$$

or with one sided free conditions such as:

$$x(0) = x_0,\ x(t_f) \text{ free.} \tag{3}$$

It is also assumed that the trajectory is bounded and takes its values in $[l_b, u_b]$. A differentiable function $x(.)$ is said to be *admissible* if it satisfies the related initial and final conditions as (2) or (3), and $x(t) \in [l_b, u_b]$ for all $t \in [0, t_f]$. Minimizing (1) may also accompanied by integral or algebraic constraints:

$$I(x,\dot{x}) = \int_0^{t_f} g(t,x(t),\dot{x}(t))dt = C_1, \tag{4}$$

$$G(x,\dot{x}) = C_2, \tag{5}$$

where $C_1$ and $C_2$ are given constants. In the case of constrained problem, an admissible function is said to be *feasible* if it satisfies (4) or (5).

The problem in its general form is infinite-dimensional. In order to handle it with digital computers, discretization or parametrization are usually applied. In the present work the discretization in time domain is proposed and introduced in the next subsection.

### 2.1 Discretization

Let assume that the time interval $[0, t_f]$ is partitioned to $N + 1$ equidistant points as:

$$\{t_0 = 0, t_1, t_2, \ldots, t_{N-1}, t_N = t_f\},$$

with $t_{k+1} - t_k = h$. Here a fixed step size is used, for optimizing the truncation error, variable step size may be also proposed like the method given in [24].

The value of unknown function and its derivative at $t_k$ are shown by $x_k$ and $\dot{x}_k$ respectively. Therefore, by using a finite difference approximation of $\dot{x}_k$, the problem in discrete form is to minimize

$$\hat{J}(x_0, \ldots, x_{N+1}) = \sum_{k=0}^{N} \hat{f}(h, k, x_k, \bar{x}_k) \tag{6}$$

where $\bar{x}_k$ is a set of $x_j$s, with $j < k$ or $j > k$ which depends on the method of finite difference. For example, if in the simplest case, the forward difference is applied, that is $\dot{x}_k \approx \frac{x_k+1-x_k}{h}$, then, $\bar{x}_k = x_{k+1}$.

Therefore, a finite-dimensional optimization problem has to be solved with $x_1, x_2, \ldots, x_{N-1}$ as its unknowns. In the fixed initial and final conditions (2), $x_0$, and $x_{N+1} = x_{t_f}$ enter the problem as constants.

In the case of free endpoint (3), $x_N$ will be add to the set of unknowns.

Integral and algebraic constraints (4)–(5) are represented similarly in discrete form, respectively as follows:

$$\hat{I}(x_0, x_1, \ldots, x_N) = \sum_{k=0}^{N} \hat{g}(h, k, x_k, \bar{x}_k) = C_1, \tag{7}$$

$$\hat{G}_k(h, x_k, \bar{x}_k) = C_2, \quad k = 0, 1, \ldots, N \tag{8}$$

In order to increase the space of feasible solutions in numerical methods, the equality conditions in (7)–(8) substitutes with inequality conditions such as

$$\left|\hat{I}(x_0, x_1, \ldots, x_N) - C_1\right| \leq \varepsilon, \tag{9}$$

$$\left|\hat{G}_k(h, x_k, \bar{x}_k) - C_2\right| \leq \varepsilon, \quad k = 0, 1, \ldots, N \tag{10}$$

where $\varepsilon$ is a small allowable tolerance.

## 2.2 Existence of the solution

The discretized form of the original problem is to minimize (6) subject to (7)–(8). The first question here is the existence of a solution for this optimization problem. In this section we try to answer this question. Let $\mathcal{P}(N)$ denotes the optimization problem consider for an arbitrary and fixed integer $N$. In the next theorem, it is proved that $\mathcal{P}(N)$ has a solution.

**Theorem 1** *If $f$, $g$ and $G_k$s are continuous, then $\mathcal{P}(N)$ has a solution.*

***Proof*** From the Weierstrass extreme value theorem [30], we know that a continuous function has minimum and maximum on compact sets. Therefore, if we could prove the continuity of $\hat{J}$ and compactness of the set of feasible solutions, then the existence of the solution will be proved. These two millstones are proved in the sequel.

*Continuity of $\hat{J}$*: As stated in (6), $\hat{J}$ is a multi-variable real valued function and is in the form of a summation. Therefore, if it is proved that a typical term in this summation, that is $\hat{f}(h, k, x_k, \bar{x}_k)$ is continuous, then it is proved that $\hat{J}$ is also continuous. $\hat{f}$ is indeed the discrete form of $f(t, x(t), \dot{x}(t))dt$ in $[t_k, t_k + 1]$. If the forward finite difference is used for derivative approximation, the discretized form is as follows:

$$\hat{f}\left(h, k, x_k, \bar{x}_k\right) = f\left(t_k, x_k, \frac{x_{k+1} - x_k}{h}\right)h \tag{11}$$

As $f$ is continuous, the right hand side of the above relation represent a continuous term. This proves that $\hat{J}$ is continuous.

*Compactness of feasible space* In the case of unconstrained problem, from the admissibility definition, the feasible space is

$$\begin{aligned}&\{(x_0, x_1, \ldots, x_N) \in \mathbb{R}^{N+1} \,:\, l_b \leq x_k \leq u_b\}\\ &\quad = \prod_{k=0}^{N} [l_b, u_b]\end{aligned} \tag{12}$$

which is compact.

In the case of constraint problem, the feasible space comes from (7)–(8). This is intersection or union of the following sets:

$$\hat{I}^{-1}([C_1 - \varepsilon, C_1 + \varepsilon]) \tag{13}$$

$$\bigcap_{k=0}^{N} \hat{G}_k^{-1}([C_2 - \varepsilon, C_2 + \varepsilon]) \tag{14}$$

As $g$ and $G_k$s are continuous, then their discretized forms are also continuous. This shows that (13) and (14) are inverse image under continuous function. On the other hand, $[C_1 - \varepsilon, C_1 + \varepsilon]$ and $[C_2 - \varepsilon, C_2 + \varepsilon]$ are closed. Therefore, the inverse image of this closed sets under continuous function are closed and finite intersection and union of compact sets are closed. As this closed sets are subsets of the compact set $\prod_{k=0}^{N} [l_b, u_b]$, they are also compact. □

## 3 Description of the proposed method

As mentioned before, WOA is a population-based metaheuristic optimization algorithm. The method has been widely used to solve optimization problems in diverse fields. Some of applications of this method are in the field of energy [4, 27], medicine [25, 34], photonics [22], structural optimization [9, 11] and fault detection [18, 38]. There are also modifications [1, 33] and hybridization of this method with other optimization methods [8]. A large number of successful implementation of WOA alongside with its continuing improvements indicate that the method has basic features and superiority in the field of meta-heuristic optimization.

In this section, WOA is applied to the problem of minimizing (6) with possibly (9) or (10) as constraints. The steps of the method are as follow:

*Initialization* At the first step, a population of whales or search agents is selected. A set of trajectory values is assigned randomly to each artificial whale in the trajectory interval $[l_b, u_b]$, such as $(x_0, x_1, \ldots, x_N)$. In the case of constrained problems, the initial trajectories are selected as feasible. Let assume that $X^i(0)$ denotes values of the trajectory assigned to $i$-th whale at the initialization step.

After initialization, the iterations start. In this phase, the values of $X^i(t)$ are updated in $t$-th iteration until convergence occurs. This step has itself some subroutines which are explained in the following.

*Finding the best solution* Among whales there is one with the best performance index. In order to find it, the corresponding derivatives of all trajectories are calculated using finite difference approximation. Then, the related performance index is determined by (6). With a simple comparing between whales' performances, the optimum will be found. Let $X^*(t)$ denotes the best solution at the $t$-th step. With this optimal solution, the other whales are updated in the next step.

*Update* The trajectory values of each whale are update by one of the two method. First a random value $p$ is generated for each whale, and then its control values are updated based on the following formula:

$$X^i(t+1) = \begin{cases} X^*(t) - A|CX^*(t) - X^i(t)|, & \text{if } p < 0.5,\ |A| \le 1 \\ D'_i d.e^{bl}.\cos(2\pi l) + X^*(t), & \text{if } p \geqslant 0.5,\ |A| \le 1 \\ X_{rand}(t) - A|CX_{rand}(t) - X^i(t)| & \text{if } p < 0.5,\ |A| > 1 \end{cases} \tag{15}$$

where

$A = 2.a.r - r$ a coefficient vector
$C = 2.r$ a coefficient vector
$a =$ a vector of constants linearly decreased from 2 to 0 during iterations
$D'_i = |X^*(t) - X^i(t)|$ the distance between the $i$-th whale and the bestsolution so far
$b =$ a constant defining the shape of the logarithmic spiral
$l =$ a random number in $[-1, 1]$
$X_{rand} =$ a randomly chosen admissible trajectory among current population

*Smoothing* An averaging process called smoothing is performed in this step on the values of each trajectory in order to smooth it by preventing from corners. If $0 < \alpha < 1$ is a weight coefficient, then as stated in [16], the proposed formula for smoothing $X^i(t+1) = (x_0, x_1, \ldots, x_N)$ at point $k$ is as follows:

$$\bar{x}_k = \begin{cases} \alpha x_{k-1} + (1-\alpha)x_{k+1}, & \text{if } \operatorname{sign}(x_k - x_{k-1})\operatorname{sign}(x_{k+1} - x_k) < 0 \\ x_k, & \text{else} \end{cases} \tag{16}$$

where $\bar{x}_k$ is the smoothed value of $x_k$ and replaced it during the next steps of iterations.

*Check feasibility* In the case of constrained problems that is in presence of (9) or (10). It is necessary to check the new solutions satisfies constraints. In this step of iterations, the feasibility of new solutions are evaluated, then feasible ones allows to enter the next iteration. However, infeasible solutions are discarded and are substituted by previous solutions. That is

$$X^i(t+1) = \begin{cases} X^i(t+1), & \text{if } X^i(t+1) \text{ is admissible} \\ X^i(t), & \text{otherwise} \end{cases} \tag{17}$$

This guarantees the feasibility of the solutions in each iteration.

The main algorithm of WOA for optimal control problem is summarized as follows:

*Choose $N$, $N_W$ (number of whales), and maximum number of iterations.*
*Determine the discretized form of the problem based on* (6), (9)-(10).
*Initialize the whales population* $X^i(0)$, $i = 1, 2, ..., N_W$ *of admissible (or feasible in the case of a constrained problem) discrete values.*
*Calculate the performance index of each search agent.*
*Find* $X^*$ *the best search agent, with* $J^*$
***while*** ($t <$*maximum number of iterations*)
  ***for*** *each search agent*
  *update* $a$, $A$, $C$, $l$, *and* $p$.
    ***if1*** ($p < 0.5$)
      ***if2*** ($|A| < 1$)
        *Update the position of the current search agent by the upper instruction of* (15).
      ***else if2*** ($|A| \geqslant 1$)
        *Select a random search agent* $X_{rand}(t)$.
        *Update the position of the current search agent according to the lower relation of* (15).
      ***end if2***
      ***else if1*** ($p \geqslant 0.5$)
        *Update the position of the current search agent by the middle instruction of* (15).
      ***end if1***
    ***end for***
  *Smooth the new candidate based on* (17).
  *Regulate the new candidate if it isn't admissible.*
  *In the case of constrained problem, check the feasibility. If the new candidate is feasible, set it as current solution. Otherwise, discard it and continue with the current solution.*
  *Calculate the performance index of each search agent.*
  *Update* $X^*(t)$ *if there is a better solution.*
  $t = t + 1$
***end while***
*return* $X^*$, *and* $J^*$

The flowchart of the proposed algorithm is also given in Fig. 1.

### 3.1 Computational complexity

In order to estimate the computational complexity of the proposed method, the upper bound of computational complexity of each functions has to be estimated. For out of loop procedures that are Discretization and Initialization, the complexity is of order $O(NN_w)$ where as mentioned before, $N$ is the number of subintervals in discretization and $N_w$ is the number of whales. For in the loop functions such as Performance Calculator, Smoothing, Regulation, Feasibility Checking, and Position Update, the complexity is also $O(NN_W)$. Therefore, the total complexity is bounded of order $O(NN_wT)$ where $T$ is the maximum number of iterations. This shows that when the number of iterations and number of whales are small with respect the number of partitions, the computational complexity of the algorithm is of order $O(N)$ and it may be encountered as a fast algorithm.

### 3.2 Programming notes

The algorithm was implemented as a computer program file with C++ programming language. For debugging and compilation, gcc compiler 5.4.0 under Linux operating system was used. A structured and user friendly program has been written. In order to work and refer to the solution of each whale, a new variable of structure type has been defined using `typedef` instruction as follows:

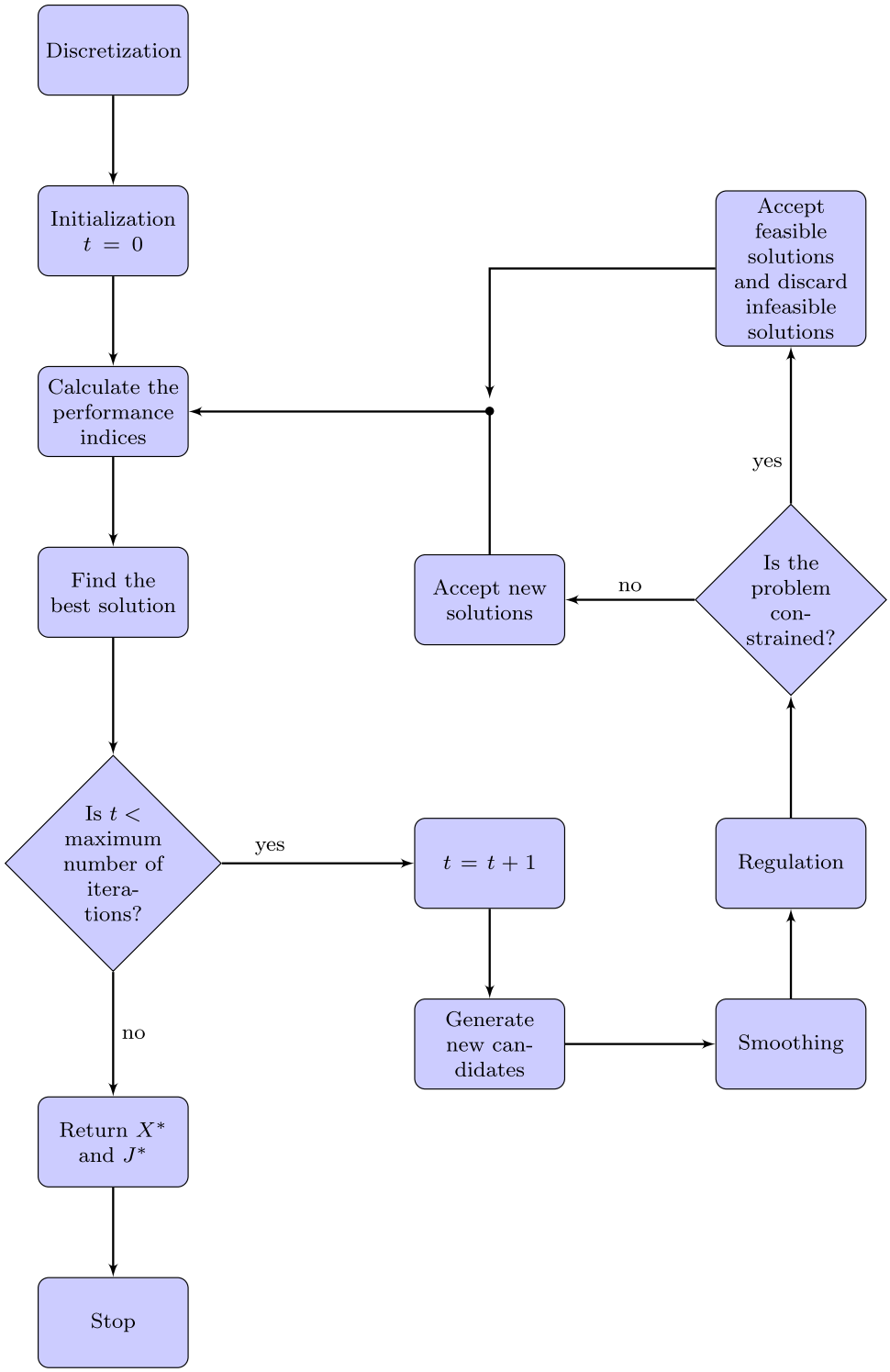


**Fig. 1** Flowchart of the proposed algorithm

**Table 1** Value of parameters in solving the problem of Example 1

| Number of whales | Maximum iterations | $l_b$ | $u_b$ | $t_f$ | $N$ | $\alpha$ |
|---|---|---|---|---|---|---|
| 100 | 150 | 0.0 | 0.5 | 1.0 | 50 | 0.5 |

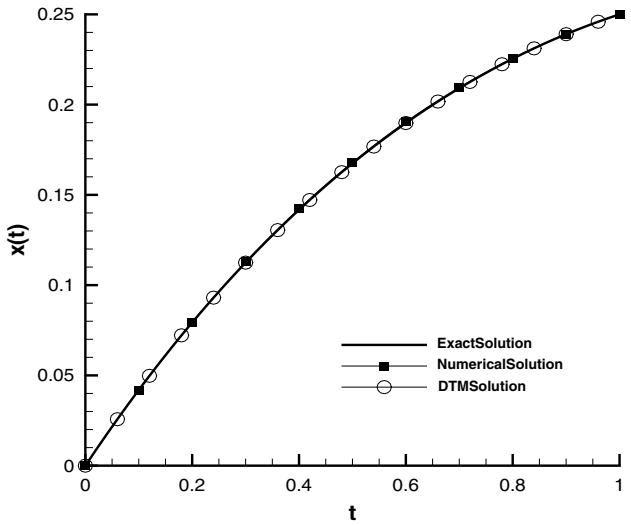


**Fig. 2** Comparison of exact, numerical and DTM solutions of Example 1 for $T_f = 1$

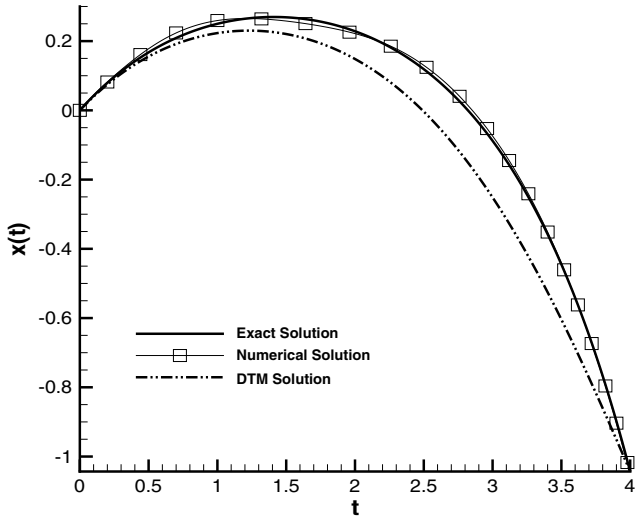


**Fig. 3** Comparison of exact, numerical and DTM solutions Example 1 for $T_f = 4$

```
typedef struct {
float positions[dim+1];
float olds[dim+1];
float xdot[dim+1];
int feasibility;
float fitness;
} whale;
```

By this definition, an individual whale has its current solution in `position` vector of floats with length of $N = \texttt{dim} + 1$. `olds` is used to hold the previous feasible solution, it substitutes `positions` when the new candidate solution isn't feasible. `xdot` contains the derivative of the current solution calculated by a difference formula. `feasibility` is an integer switch (0 or 1) indicating that the new candidate is feasible or not. The performance index of the current solution is referred by `fitness`.

As there are a number of whales as search agents, a vector of `whale` variable with `sample` as variable name is defined as:

`whalesample[SearchAgents_no];`

All stages of the algorithm or blocks in flowchart are implemented in structured format as function. Some of the most important ones are listed as follow:

`void` initialization(`void`): In this function an initial (feasible) solution is generated and assign to each whale.

`void` smoothing(`int`): The solutions are smoothed by reduction of corner points in this function.

`void` regulation(`void`): Admissibility is checked and out of bound values are resolved.

`void` position_update(`void`): The current solution of each $t$-th whale is stored and `sample[t].old`, and new solution is generated based on WOA algorithm and stored temporary in `sample[t].positions`.

`void` FeasibilityEvaluation(`int`): Using this function, the feasibility switch of each whale takes its value, 0 for infeasible and 1 for feasible.

`float` IndexEvaluation(`int`): Derivatives and performance index are evaluated by this function

`void` check_feasibility(`void`): For infeasible solutions the new found solutions (`sample[t].positions`) are discarded and is substitute by the current one (`sample[t].olds`).

`void` finding_local_opt(`void`): This function finds the whale with optimal fitness by comparing all solutions with current best solution.

`void` MakeInput4TecPlot(`char*`,`char*`,`float*`,`float*`): In order to prepare output for drawing by a standard plotting software, this function has been developed. It makes a file form outputs for using in Tecplot® software. Its inputs are file names as two string of characters, the vector of optimal solutions and the performance indices of each iteration.

In the next section, the method is applied on four examples to evaluate its performance and accuracy. It is to be

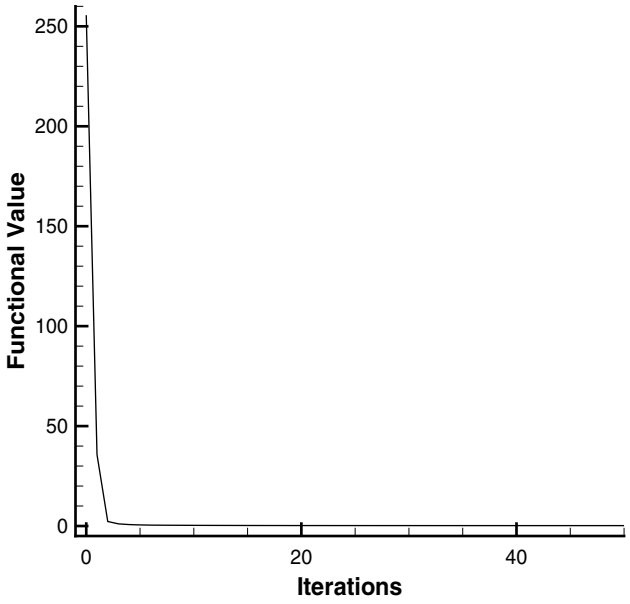


**Fig. 4** The functional value of Example 1 versus iterations number

noted that the source codes are publicly available at http://www.ari.ac.ir/code/SWOACVP.rar.

## 4 Numerical examples

In this section, the proposed method is evaluated by applying it on some practical and well known examples.

***Example 1*** Let consider the problem of minimizing

$$J(x,\dot{x}) = \int_0^1 \left(x^2 + t\dot{x} + \dot{x}^2\right)dt \tag{18}$$

subject to $x(0) = 0$, $x(1) = 0.25$ ([14]). This problem has the following exact solution:

$$x^*(t) = 0.5 + \frac{2-e}{4(e^2-1)}e^t + \frac{e-2e^2}{4(e^2-1)}e^{-t} \tag{19}$$

The problem was solved by the proposed method with parameters as shown in Table 1.

The resulting solution has been compared with the exact one, that is in Fig. 2. It is clear that the numerical solution approximated the exact solution with reasonable accuracy. In order to compare with another numerical method, the DTM solution with degree 7 was also obtained that is

$$\begin{aligned} x(t) =& 0.44378t - 0.25000t^2 + 0.07396t^3 - 0.02080t^4 \\ &+ 0.00370t^5 - 0.00069t^6 + 0.00009t^7 \end{aligned}$$

It is evident that the proposed method and DTM lead to results coincide the exact one. However, DTM usually is restricted to short time ranges because of validating Taylor series near the initial point. To check this point, the problem was solved for a final time over $t = 1$. For example, $T_f = 4$ was chosen with final condition $x(T_f) = -1.0432$, then the previous exact solution remains the optimal trajectory. The DTM solution with the same degree in this case is

$$\begin{aligned} x(t) =& 0.420954t - 0.25000t^2 + 0.07016t^3 - 0.02080t^4 \\ &+ 0.00351t^5 - 0.00069t^6 + 0.00008t^7 \end{aligned}$$

The result of the proposed method is compared with the DTM and exact solution, in Fig. 3. As it could be seen, the proposed numerical method has a result with high accuracy, while the DTM solution has considerable drift when it passed $t = 1$. This shows that the proposed method isn't restricted to short ranges and is more practical in problems with long time ranges with respect to DTM which is valid only near the initial point.

Being a criterion of convergence rate, the value of functional at each iteration is also depicted in Fig. 4. It shows that the method converges in early iterations and has a high rate of convergence.

To make a comparison of the proposed method with other evolutionary schemes, an error function has been defined as follows:

$$\text{Error} = \frac{1}{N+1}\sum_{i=0}^{N+1} |x^*(t_i) - x(t_i)| \tag{20}$$

where $x^*(t_i)$ and $x(t_i)$ are the value of the optimal solution and the related numerical solution at $t_i$. The error is an

**Table 2** Comparing the proposed method with other numerical algorithms

| | PSO | | GWO | | MFO | | WOA | |
|---|---|---|---|---|---|---|---|---|
| | Error | Time | Error | Time | Error | Time | Error | Time |
| Run 1 | 0.0026 | 0.5842 | 0.0023 | 0.5971 | 0.0041 | 0.5608 | 0.0006 | 3.8717 |
| Run 2 | 0.0023 | 0.5520 | 0.0028 | 0.5793 | 0.0017 | 0.5582 | 0.0007 | 3.8285 |
| Run 3 | 0.0023 | 0.5209 | 0.0015 | 0.5804 | 0.0043 | 0.5683 | 0.0005 | 3.8187 |

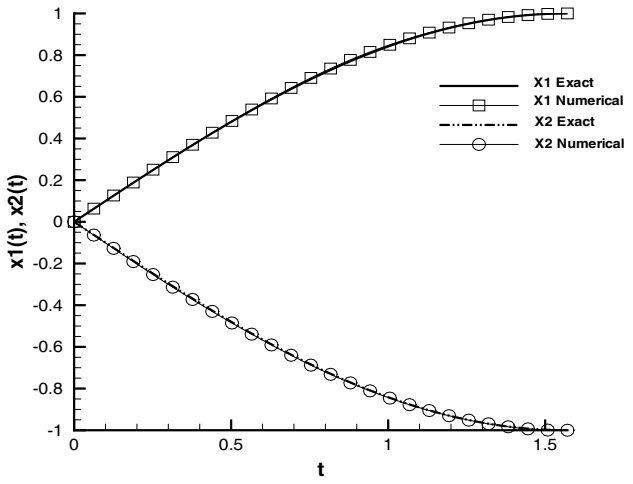


**Fig. 5** The exact and numerical solutions of Example 2 for $T_f = \pi/2$

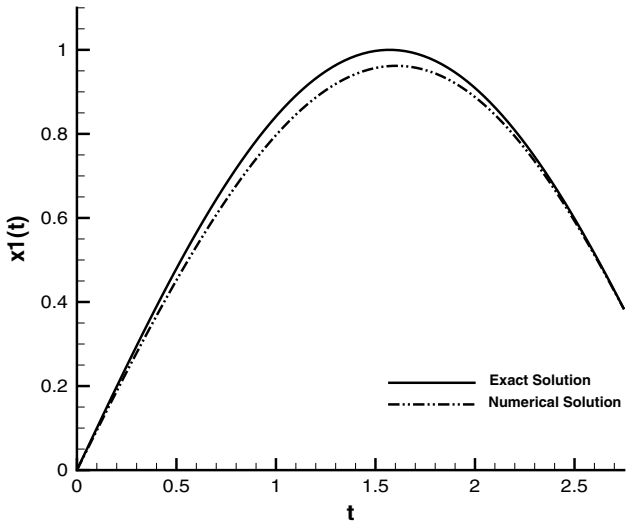


**Fig. 6** The exact and numerical solutions of Example 2 for $T_f = 7\pi/8$

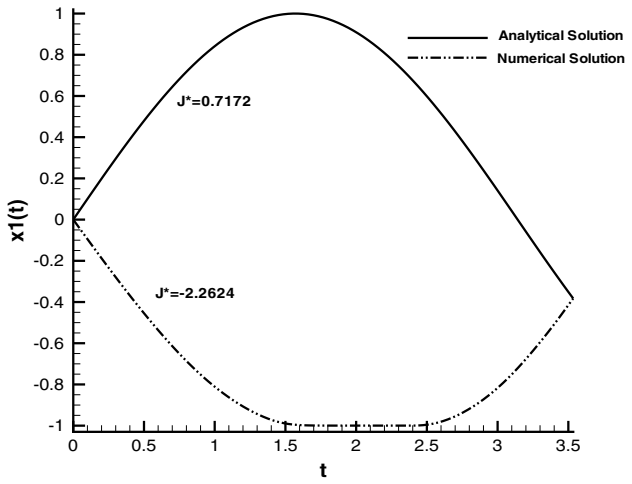


**Fig. 7** The analytical and numerical solutions of Example 2 for $T_f = 9\pi/8$

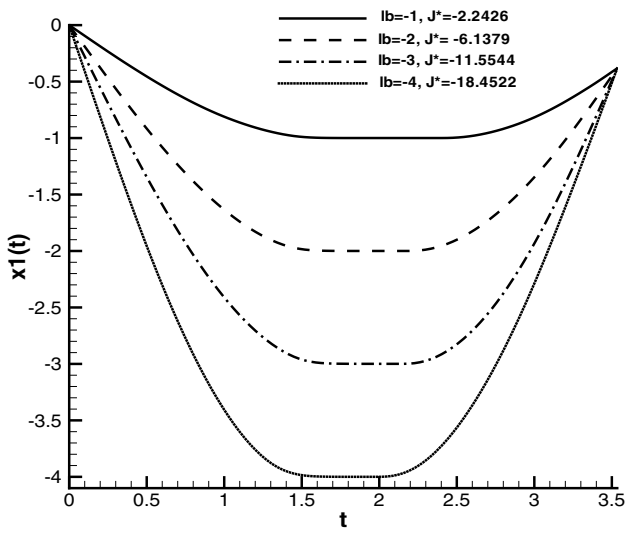


**Fig. 8** Numerical solutions of Example 2 for various lower bounds

averaged norm indicating the compliance of the numerical solution with the exact one.

The results of Example 1 has been obtained by implementing the Particle Swarm Optimization (PSO), Gray Wolf Optimizer (GWO) [19], and Moth Flame Optimization algorithm (MFO) [20]. Method's parameters for all of these algorithms are the same as Table 1. Every algorithm was executed three times and the resulting error and time of execution in seconds were given in Table 2.

Comparing the results shows that the WOA method proposed in this paper leads to more accurate solutions of order $10^{-4}$, while the other methods have less accuracy of order $10^{-3}$. On the other hand, the time of PSO, GWO and MFO are approximately the same, while the WOA method needs more execution time. However, the relative increase in time is about 6.7 that leads to 10 times improving the accuracy. Therefore, the proposed method has an accuracy to time superiority about 1.49 with respect to the mentioned algorithms.

***Example 2*** In order to check the effect of smoothing on oscillating solutions, in the second example a problem with sin-wave form solutions is studied. Consider the minimization of the following functional

$$J(x_1, \dot{x}_1, x_2, \dot{x}_2) = \int_0^1 \left(\dot{x}_1^2 + \dot{x}_2^2 + 2x_1x_2\right)dt \tag{21}$$

subject to

$$x_1(0) = 0 \;\; x_2(0) = 0 \tag{22}$$

$$x_1(\pi/2) = 1 \;\; x_2(\pi/2) = -1 \tag{23}$$

$x(0) = 0$, $x(1) = 0.25$ ([26]). This problem has the following exact solution which are obtained using Euler–Lagrange equation:

$$x_1(t) = \sin(t), \;\; x_2(t) = -\sin(t) \tag{24}$$

The problem was solved by the proposed method, and the results are given in Fig. 5. It is clear that the resulting curves are accurate and approximate the exact analytical curves as well. As the exact solutions are sinus forms, it is interesting to check whether the numerical solutions track the exact ones or not when the time period increases from $\pi/2$ to $\pi$. Therefore, in the second group of runs, the final time was increased from $\pi/2$ to near $\pi$. The numerical solution for these case studies approximated the analytical solution as well. For example, Fig. 6 shows the result for $T_f = 7\pi/8$. Because of symmetry, $x_2(t)$ has not been depicted. It deduced from this result that the smoothing process does not annihilate the increasing-decreasing form of solutions. Therefore, the method is not restricted to monotonic functions and could be applied to problems with general solutions.

When $T_f$ reaches to $\pi$, the optimal solution switches to the trivial solution $x_1(t) = x_2(t) = 0$. To check the solution $T_f > pi$, the exact analytical solutions has sinus form and obtained from Euler–Lagrange. As an example, the problem was solved for $T_f = 9\pi/8$ with final conditions as

$$x_1(T_f) = -0.3827, \;\; x_2(T_f) = 0.3827$$

Then the exact solution remains as previous $x_1^*(t) = \sin(t)$, $x_2^*(t) = -\sin(t)$. The numerical results, however changes the behavior for this example as shown in Fig. 7. Moreover, the objective value for the numerical case is $I^* = -2.2624$ which is lower than the analytical solution that is $I^* = 0.7172$. The reason is that the exact solution (24) was obtained from solving the related Euler–Lagrange equation which requires that the solution has forth order continues derivatives. Indeed, this exact solution is valid in the class of $C^4$ functions on $[0, T_f]$. In the numerical scheme, on the other hand, the approach and smoothing only guarantee first order continues derivatives, that is it searches for the minimizer in a bigger class of $C^1$ functions. Therefore, by using the proposed method, more accurate optimal solutions may be found with respect to indirect methods using Euler–Lagrange equation. The value of the solution in Fig. 7 is restricted to $[-1, 1]$. If these bounds change, the method leads to solutions with lower objective values. Figure 8 shows some of them calculated for various control bounds: $l_b = -1, -2, -3, -4$. The values of the related objective function show that it is a decreasing function of $l_b$.

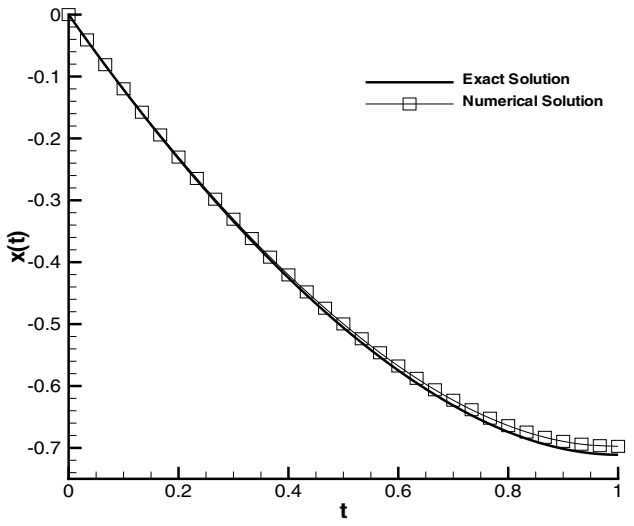


**Fig. 9** The exact and numerical solutions of Example 3

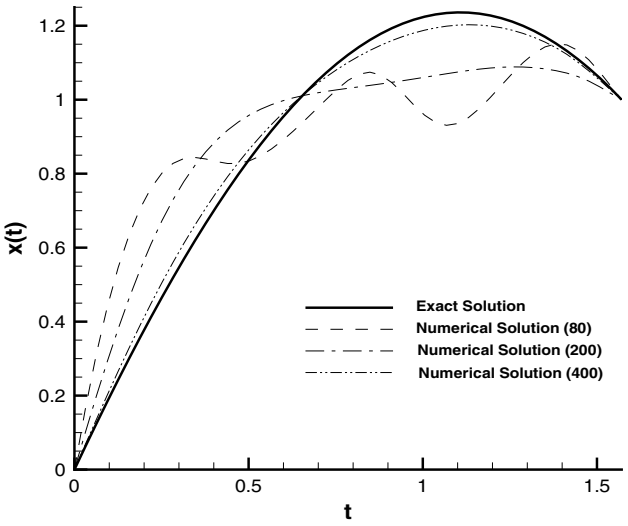


**Fig. 10** Numerical solutions of Example 4 for various iterations

**Table 3** Value of parameters in solving the problem of Example 4

| Number of whales | Maximum iterations | $l_b$ | $u_b$ | $t_f$ | $N$ | $\alpha$ | $\varepsilon$ |
|---|---|---|---|---|---|---|---|
| 20 | 80,200,400 | 0.0 | 1.3 | $\pi/2$ | 50 | 0.5 | 0.05 |

***Example 3*** In this example, the method is examined in a case with a free endpoint. Let consider the problem of minimizing

$$J(x,\dot{x}) = \int_0^1 \left(x^2 + \dot{x}^2 + 2tx\right)dt \tag{25}$$

accompanied with initial condition $x(0) = 0$ and free endpoint condition. This problem has the following exact analytical solution resulting from Euler–Lagrange equation and Transversality condition.

$$x^*(t) = -\frac{e}{e+e^{-1}}e^t + \frac{e}{e+e^{-1}}e^{-t} + \frac{1}{2}te^t \tag{26}$$

The problem was solved by the proposed numerical method without any modification except removing the final condition from calculations. Being the unknown final condition causes a miss distance at the endpoint as it is inferred from Fig. 9. The miss distance in this example is 0.0136 which confirms the accuracy of the method. The objective was $I^* = 0.8565$ which is near the analytical one that is $I^* = 0.9283$.

Therefore, the method works well in cases with free final conditions.

***Example 4*** In the last example, a constrained problem is considered. The aim is to minimize

$$J(x,\dot{x}) = \int_0^{\pi/2} \left(\dot{x}^2 - x^2\right)dt \tag{27}$$

subject to

$$\int_0^{\pi/2} 2x dt = 6 - \pi \tag{28}$$

with initial and final condition $x(0) = 0$, $x(\pi/2) = 1$. The analytical solution using Euler–Lagrange equation is $x^*(t) = 2\sin x + \cos x - 1$, with objective $J^* = 2 - \frac{\pi}{2}$. The problem has been solved by the method with parameters in Table 3. In order to check the convergence of the numerical solution to the exact one, the results of each iteration is compared with the exact solution. For example, in the Fig. 10, the resulting curves after of 80, 200 and 400 iterations were drawn and compared with the exact one. It can be seen that the result after 80 iterations differs from the exact solution. However, in subsequent iterations the error of the numerical results is reduced and after 500 iterations, the numerical and analytical solutions coincide. This shows that the WOA method works well in finding the optimum solution in the case of constrained problems too.

Taken together, the results showed the superior performance of the proposed WOA-based numerical method. One of the main reasons is due to the high exploratory behavior of WOA. As a stochastic algorithm, WOA has been equipped with several random components that combine the current solutions to produce new ones. This assists this algorithm to show high exploration and provides extensive coverage of a search space. However, the search is slowed down by adaptive mechanisms which results in more local search proportional to the number of iterations.

## 5 Conclusion

A numerical method for solving calculus of variations problem was presented. The proposed method utilizes the whale optimization algorithm for optimization. The method was implemented on a batch of examples to show the ability to solve problems with different dimensions and endpoint type. Results indicate that the method is accurate and efficient. Numerical results compared differential transform method reveals that the method has some benefits and features such as:

- Unlike the DTM, the proposed method solves the constrained problems.
- The method isn't restricted to fixed endpoint problems.
- In contrast with indirect methods, solving problems with solutions having discontinuous derivatives was experienced

For further works, implementing the method on computers with GPU and evaluating its parallel performance is suggested.

## References

1. Abdel-Basset M, El-Shahat D, Sangaiah AK (2019) A modified nature inspired meta-heuristic whale optimization algorithm for solving 0–1 knapsack problem. Int J Mach Learn Cybern 10(3):495–514
2. Abdulaziz O, Hashim I, Chowdhury MSH (2008) Solving variational problems by homotopy perturbation method. Int J Numer Methods Eng 75(6):709–721
3. Aubert G, Vese L (1997) A variational method in image recovery. SIAM J Numer Anal 34(5):1948–1979
4. Chen Y, Vepa R, Shaheed MH (2018) Enhanced and speedy energy extraction from a scaled-up pressure retarded osmosis process with a whale optimization based maximum power point tracking. Energy 153:618–627
5. Dehghan M, Tatari M (2006) The use of Adomian decomposition method for solving problems in calculus of variations. Math Probl Eng 2006:1–12

6. Ghasemi M (2016) On using cubic spline for the solution of problems in calculus of variations. Numer Algorithms 73(3):685–710
7. Guo K, Liu B, Li X, Liu H, Liu C (2016) Flow pattern construction-based tubular heat transfer intensification using calculus of variations. Chem Eng Sci 152(2):568–578
8. Jadhav PP, Joshi SD (2018) Fractional weightage based objective function to a hybrid optimization algorithm for model transformation. Evol Intell. https://doi.org/10.1007/s12065-018-0179-8
9. Kamgar R, Khatibinia M, Khatibinia M (2019) Optimization criteria for design of tuned mass dampers including soil–structure interaction effect. Int J Optim Civil Eng 9(2):213–232
10. Kamien ML, Schwartz NL (2012) Dynamic optimization: the calculus of variations and optimal control in economics and management, 2nd edn. Dover Publications, New York
11. Kave A, Ilchi Ghazaan M (2017) Enhanced whale optimization algorithm for sizing optimization of skeletal structures. Mech Based Des Struct Mach: Int J 45(3):345–362
12. Komzsik L (2009) Applied calculus of variations for engineers. CRC Press, Boca Raton
13. Levin Y, Nediak M, Ben-Israel A (2002) A direct Newton method for calculus of variations. J Comput Appl Math 139(2):197–213
14. Maleki M, Mashali-Firouzi M (2010) A numerical solution of problems in calculus of variation using direct method and nonclassical parameterization. J Comput Appl Math 234(5):1364–1373
15. Malinowska AB, Torres DFM (2010) Leitmann's direct method of optimization for absolute extrema of certain problems of the calculus of variations on time scales. Appl Math Comput 217(3):1158–1162
16. Mehne HH, Mirjalili S (2018) A parallel numerical method for solving optimal control problems based on whale optimization algorithm. Knowl-Based Syst 151(1):114–123
17. Meucci A, Nicolosi M (2016) Dynamic portfolio management with views at multiple horizons. Appl Math Comput 274:495–518
18. Miao Y, Zhao M, Makis V, Lin J (2019) Optimal swarm decomposition with whale optimization algorithm for weak feature extraction from multicomponent modulation signal. Mech Syst Signal Process 122:673–691
19. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61
20. Mirjalili S (2015) Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm. Knowl-Based Syst 89:228–249
21. Mirjalili S, Lewis A (2016) The whale optimization algorithm. Adv Eng Softw 95:51–67
22. Mirjalili SZ, Mirjalili SM, Saremi S, Mirjalili S (2020) Whale optimization algorithm: theory, literature review, and application in designing photonic crystal filters. In: Mirjalili S, Song Dong J, Lewis A (eds) Nature-inspired optimizers. studies in computational intelligence, vol 811. Springer, Cham
23. Mohammadi R, Shamsyeh Zahedi M, Bayat Z (2015) Numerical solution of calculus of variation problems via exponential spline method. Math Sci Lett: Int J 4(2):101–108
24. Morrison D (1962) Optimal mesh size in the numerical integration of an ordinary differential equation. J ACM 9(1):98–103
25. Mostafa A, Hassanien AE, Houseni M, Hefny H (2017) Liver segmentation in MRI images based on whale optimization algorithm. Multimed Tools Appl 76(23):24931–24954
26. Nazemi AR, Hesam S, Haghbin A (2013) A fast numerical method for solving calculus of variation problems. Adv Model Optim 15(2):133–149
27. Oliva D, Abd El Aziz M, Ella Hassanien A (2017) Parameter estimation of photovoltaic cells using an improved chaotic whale optimization algorithm. Appl Energy 200:141–154 (author links open overlay panel)
28. Razzaghi M, Marzban HR (2000) A hybrid analysis direct method in the calculus of variations. Int J Comput Math 75(3):259–269
29. Razzaghi M, Yousefi S (2001) Legendre wavelets method for the solution of nonlinear problems in the calculus of variations. Math Comput Modell 34(1–2):45–54
30. Rudin W (1976) Principles of mathematical analysis. McGraw Hill, New York City
31. Saadatmandi A, Dehghan M (2008) The numerical solution of problems in calculus of variation using Chebyshev finite difference method. Phys Lett A 372(22):4037–4040
32. Sadiku MNO (2001) Numerical techniques in electromagnetics (second edition), chapter 4. CRC Press, Boca Raton
33. Sun Y, Wang X, Chen Y, Liu Z (2018) A modified whale optimization algorithm for large-scale global optimization problems. Expert Syst Appl 114:563–577
34. Tharwat A, Moemen YS, Hassanien AE (2017) Classification of toxicity effects of biotransformed hepatic drugs using whale optimized support vector machines. J Biomed Inform 68:132–149
35. Venugopal N, Grandhi RV, Hankey WL, Belcher PJ (1991) Automated trajectory synthesis for hypersonic vehicles using energy management and variational calculus techniques. Acta Astronaut 25(11):669–678
36. Yousefi SA, Dehghan M (2010) The use of He's variational iteration method for solving variational problems. Int J Comput Math 87(6):1299–1314
37. Zarebnia M, Birjandi M (2012) The numerical solution of problems in calculus of variation using B-spline collocation method. J Appl Math 2012, 605741. https://doi.org/10.1155/2012/605741
38. Zhang X, Liu Z, Miao Q, Wang L (2018) Bearing fault diagnosis using a whale optimization algorithm-optimized orthogonal matching pursuit with a combined time–frequency atom dictionary. Mech Syst Signal Process 107:29–42