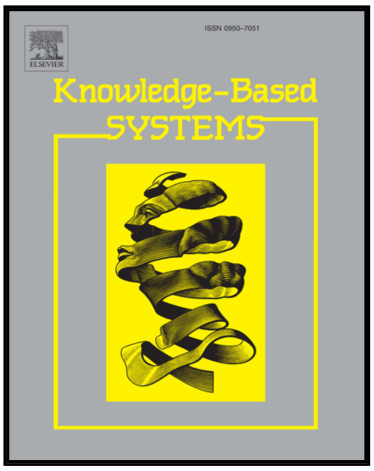


A Parallel Numerical Method for Solving Optimal Control Problems based on Whale Optimization Algorithm

Hamed Hashemi Mehne, Seyedali Mirjalili

PII: S0950-7051(18)30156-4
DOI: 10.1016/j.knosys.2018.03.024
Reference: KNOSYS 4273

To appear in: *Knowledge-Based Systems*

Received date: 17 October 2017
Revised date: 16 March 2018
Accepted date: 17 March 2018

A Parallel Numerical Method for Solving Optimal Control Problems based on Whale Optimization Algorithm

Hamed Hashemi Mehne[a], Seyedali Mirjalili[b]

[a]*Aerospace Research Institute, 14665-834, Tehran, Iran*
[b]*School of Information and Communication Technology, Griffith University, Nathan Campus, Brisbane, QLD 4111, Australia*

Abstract

Development of a numerical algorithm for solving optimal control problems is reported in this article. The method is a combination of multi-staging of the original problem to a finite dimensional optimization problem and the recently proposed Whale Optimization Algorithm (WOA). The method is proposed to reduce the required number of iterations. The parallel implementation is also proposed and discussed. Numerical examples are given to check the validity and accuracy of the proposed method. Results show that method converges to the exact solution with accuracy comparable to other numerical methods.

Keywords: Optimal control, Whale optimization algorithm, Numerical solution, Parallel computing

1. Introduction

Optimal control problems have a wide rage of real world application and are usually challenging due to their large scale and complexity of relations. Therefore, finding the analytical solutions by traditional method for such problems have a little chance of success. Naturally, with the development of computers, many researches have focused on numerical methods for solving optimal control problem. These algorithms have been pre-

URL: `hmehne@ari.ac.ir` (Hamed Hashemi Mehne), `seyedali.mirjalili@griffithuni.edu.au` (Seyedali Mirjalili)

sented on a wide spectrum of algorithms with different mathematical bases such as control parametrization [7], iterative dynamic programming [9], state parametrization[10], and data-based approximate policy [8].

In recent years, evolutionary and heuristic optimization algorithms are in the center of attention for solving optimal control problems based on their independency to derivatives and high rate of convergence. Some of application of these to optimal control problems are listed and reviewed below:
Genetic Algorithm: genetic algorithm (GA) is a meta-heuristic evolutionary algorithm for optimization inspired by the process of natural selection. The first application of genetic algorithm to optimal control problem was presented in [12] where discrete-time optimal controls were under consideration. It followed by a modified genetic algorithm for discrete-time optimal control problems which was presented in [13]. It benefits form floating point representation and specialized operators to enhance more accurate solutions with respect to classical GA implementation on optimal control problem. GA was also used in the method of [18] as one of the iterations step for solving a special class of optimal control problem. GA tends to guess initial solution for the nonlinear programming step. The method is restricted to dynamical systems with controls appearing only linearly. The numerical solution of optimal control problem with terminal constraints and singular inputs using GA was proposed in [24]. The method uses spline interpolation for control parametrization. For recent applications of GA in optimal control, the reference [15] may be addressed where a modified genetic algorithm is drived for solving non-linear optimal control problems numerically. The underlying system is subjected to equality, inequality and final state conditions.
Particle Swarm Optimization: particle swarm optimization (PSO) is another meta-heuristics and belongs to the category of swarm-based optimizers. One of the First attempts of exploiting PSO in optimal control problems is reported in [4], where hybrid autonomous switched systems were under consideration. PSO was used to find time instance of switches that minimizes a performance index. Optimal control of switched linear systems was discussed in [5] by using PSO where switching time instants and continuous input have been determined simultaneously. Continuous control functions are obtained from Hamilton-Jacobi-Bellman as state feedback control laws. The method is restricted to linear dynamical systems. More general optimal control problems were studied with PSO in [2], where the original problem was converted to a quasi assignment problem by partitioning the control set. Then the problem is to assign the best control value into a the time subsection which has

been solved numerically by PSO.

Invasive Weed Optimization: invasive weed optimization (IWO) is an ecologically inspired meta-heuristic-optimization technique, mimicking the ecological behavior of colonizing weeds. The pioneer work of applying IWO to optimal control problems is [2]. In the approach of [2], the set of control values is divided into a partition and then by defining binary decision variables the problem is converted to a quasi-assignment problem which is solved by IWO. Another example is [6], where IWO was utilized to solve the control problems. In order to improve the accuracy of parametrization as piecewise constant functions, Bezier curves are used in parametrization. Despite of obtaining smooth solutions, the Bezier curve approach increases the computational complexity and makes the method time consuming with large scale optimal control problem.

Ant Colony Optimization: ant colony optimization (ACO) is a swarm-based meta-heuristic algorithm for large scale optimization. The first use of this method for solving optimal control problem was suggested in [3]. The method of discretization of control function into piecewise constant forms is similar to [2]. Then the ACO is used to solve the resulting quasi-assignment problem. Further works was documented in [22], where ACO has been implemented to design state feedback controllers. Parallel processing version of the method of [3] was also introduced in [11] for message passing and for shared memory systems.

Differential Evolution: differential evolution is also an evolutionary optimization algorithm. DE was applied for solving complex and high dimensional optimal control problems in chemical process [19]. The search for optimal control values is performed in this method within a set of discrete values.

Despite of convergence rate, the most of the above mentioned meta-heuristic approaches have some drawbacks:

1- As they need to integrate obtained control function in time steps, they miss the final conditions because of drifts ([3, 2] for example).

2- They lead usually to fluctuating controls that is a function with many switches between control values, which is not suitable in real world implementation ([3, 2] for example).

3- They restricted to some special forms of dynamical systems or input signals ([18, 22, 4, 5] for example).

In the present paper, a direct numerical method for solving optimal control problems is proposed. The method is based on discretizing the dynamical system in time interval. This is a well-known method of multi-staging that converts the original problem to an optimization problem with control values as the main unknowns. To solve the resulting optimization problem, the recently introduced method of whale optimization algorithm (WOA) [14] is used. WOA is a nature-inspired meta-heuristic algorithm that simulates the social behavior of humpback whales in order to solve optimization problems. The hybrid method of multi-staging and WOA is proposed and improved by a smoothing process that reduces the computational complexity. Moreover, the parallel nature of the WOA motivates to develop the parallel version. Parallel processing approach of the method reduces the time of execution and makes the method more practical for running on shared memory computers.

The method is applied on some real world examples in order to evaluate the performance, accuracy, and implementation.

The results have negligible miss distances at final state. Due to its continues search for control values, the proposed method finds more smooth solutions in comparison with methods that search for control values in discrete mode. The type of the underlying dynamical system does not affect of the method and even complex non-linear systems can be treated. Therefore, the above mentioned triple drawbacks do not include in the proposed method.

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 Section 3. Section 4 provide results and relevant discussions. Finally, the conclusions and future works are given in Section 5.

2. Problem Statement and Discretization

For this study, a general optimal control problem with nonlinear dynamics is considered. The aim is to control the system from an initial state to a final desirable state in an optimum way which covers many practical problems. The problem may be formulated as minimizing a functional as

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

where, $u(t)$ is the control function, $x(t)$ is the related response and J is the performance index. The control state function are respectively input and response of the following dynamical system:

$$\dot{x}(t) = g(t, x(t), u(t)). \tag{2}$$

Initial and final conditions are also given as:

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

Let assume, for simplicity, that the control function $u(.)$ is single real valued. The extension to multi input systems is straightforward.
The control function is said to be *admissible* when it takes its values in $[l_b, u_b]$.

2.1. Multi-staging

Let assume that the time interval $[0, t_f]$ is divided to N subinterval:

$$[0, t_1], [t_1, t_2], \cdots, [t_{N-1}, t_N]. \tag{4}$$

If the control function takes a constant value such as u_k in $[t_{k-1}, t_k]$, then the corresponding response to u_k can be obtained by using u_k, and the pervious values of control and state, that is

$$\{(x_j, u_j) : j = 0, 1, \cdots, k-1\}. \tag{5}$$

In multi-staging, the time subinterval $[t_{k-1}, t_k]$ is also divided to $P+1$ points such as $\{t_{k_0} = t_{k-1}, t_{k_1}, \cdots, t_{k_P} = t_k\}$. Then, the corresponding states will be calculated via a numerical method for initial value problems, such as Runge-Kutta [17]. This may be interpreted as

$$x_{k_j} = G(t_{k_j}, \bar{x}_{k_j}, u_k), \quad j = 1, 2, \cdots, P, \tag{6}$$

where,

$$\bar{x}_{k_j} = \begin{cases} x_k, \quad \text{if} \ \ j = 0 \\ \\ \{x_{k_i}, i = 0, 1, 2, \cdots, j-1\}, \quad \text{if} \ \ 1 \leqslant j \leqslant P \end{cases} \tag{7}$$

The performance index is also discretized as the same way:

$$J(u_1, u_2, \cdots, u_N) = \sum_{k=0}^{N} \sum_{j=0}^{P} F(t_{k_j}, x_{k_j}, u_k). \tag{8}$$

Therefore, J is a function of the control values u_k restricted to lower and upper bounds as:

$$l_b \leqslant u_k \leqslant u_b, \quad k = 0, 1, \cdots, N. \tag{9}$$

The optimum values of $u_0, u_1, \cdots, u_N$ determine the solution of the problem approximately. Therefore, the optimal control problem (1)-(3) is changed to minimization of (8) subject to (6) and (9). This is a finite dimensional optimization problem with $\{u_0, u_1, \cdots, u_N\}$ as unknowns.
In contrast with methods of [2, 10] where the control interval $[l_b, u_b]$ is partitioned, in the present approach, a decision variable u_k is free to choose any value in $[l_b, u_b]$. This provides a continuous search for minimizers with less computational complexity instead of a discrete search arising from assignment problem.

2.2. Solvability

The problem of minimizing (8) subject to (9) is a finite dimensional optimization problem on a subset of $\mathbb{R}^N$, for fixed values of N and P. This problem is expressed as

$$\min\{J(u_1, u_2, \cdots, u_N) : l_b \leqslant u_k \leqslant u_b, \quad k = 0, 1, \cdots, N\}. \tag{10}$$

The first step in driving numerical method to solve this problem is to assure about the solvability or existence of a solution. As the set of arguments,

$$U(N) = \{(u_1, u_2, \cdots, u_N) : l_b \leqslant u_k \leqslant u_b, \quad k = 0, 1, \cdots, N\} \tag{11}$$

is a compact set in $\mathbb{R}^N$, based on the extreme value theorem, if J is continuous on $U(N)$, then is attains its minimum on this set, that is the problem is solvable. Therefore, it is required to look for conditions under which the function J is continuous. As given in (8), J is a double summation of $F(t_{k_j}, x_{k_j}, u_k)$. So, it is sufficient to prove the continuity of $F(t_{k_j}, x_{k_j}, u_k)$. As this term comes from discretizing the integral of (8), it is a constant multiplier of $f(t_{k_j}, x_{k_j}, u_k)$. The function f has tree arguments, the first one is the time and the second is the state value of the system which depends implicity to the third argument that is the control value. Therefore, if the function f is continuous on the set of related values and the state is also continuous function of control, then the continuity of J is proved and it will have a minimizer in $U(N)$. The continuity of f has to be an assumption of the problem, however, the continuity of x_{k_j} as a function of $(u_1, u_2, \cdots, u_N)$ will be proved in the sequel.

Theorem 1. *If the system describing function g in* (2) *is continuous, then the state vector of the system which is obtained form* (6) *is a continuous function of* $(u_1, u_2, \cdots, u_N)$.

Proof: In order to use induction, let consider the following renumbering of x_{k_j}:

$$x_q = x_{k_j}, \quad \text{where } q = (k-1) * (P+1) + j. \tag{12}$$

Most of the numerical methods for solving ordinary differential equations used in driving (6) expressed the current value of the solution in terms of the pervious value plus a weighted average of derivatives in current step. For example, in fourth order Runge-Kutta formula is written as:

$$\begin{aligned} x_{q+1} &= x_q + \frac{h}{6}k_1 + \frac{h}{3}k_2 + \frac{h}{3}k_3 + \frac{h}{6}k_4 \\ k_1 &= g(t_q, x_q, u_q) \\ k_2 &= g(t_q + \frac{h}{2}, x_q + \frac{k_1}{2}, u_q) \\ k_3 &= g(t_q + \frac{h}{2}, x_q + \frac{k_2}{2}, u_q) \\ k_4 &= g(t_q + h, x_q + k_3, u_q). \end{aligned}$$

Therefore, in a general format, the state vector components can be expressed as:

$$x_{q+1} = x_q + \sum_{s=0}^{S} \alpha_s g(t_q + h_s, x_q + k_s, u_q), \tag{13}$$

where, h_ss are constant and k_ss depend to the current value of g.
Now, for $q = 1$ we have

$$x_1 = x_0 + \sum_{s=0}^{S} \alpha_s g(t_0 + h_s, x_0 + k_s, u_0). \tag{14}$$

As g is continuous in all of its arguments, and x_0 is fixed, then the right hand side of (14) is a continuous function of $(u_1, u_2, \cdots, u_N)$. So, x_1 is continuous as a function of the control value. Now, let the q-th component of the state vector (x_q) is continuous, then the right hand side of (13) consists of summations and combinations of values of x_q and g at x_q. Based on continuity of g and x_q, therefore, the right hand side of (14) is continuous and then x_{q+1} is also a continuous function of $(u_1, u_2, \cdots, u_N)$. This proves the theorem by induction. $\square$

Briefly, when the system describing function and cost function's integrand are continuous, then the modified problem in finite dimension has a solution.

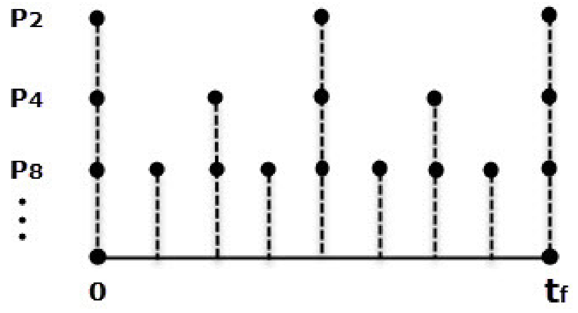


Figure 1: An example of nested partitions.

2.3. Convergence

In order to investigate the convergence of the solution of minimizing (8) subject to (9) to the solution of the original problem of minimizing (1) subject to (2)-(3), let define:

$$J_N^* = \min\{J(u_1, u_2, \cdots, u_N) : l_b \leqslant u_k \leqslant u_b, \quad k = 0, 1, \cdots, N\}. \tag{15}$$

J_N^* is indeed the optimal value of the discrete problem corresponding to the following partition of the time interval:

$$P_N = \{t_0, t_1, \cdots, t_N\}. \tag{16}$$

If J^* is the optimal value of minimizing solution of (1) subject to (2)-(3), then we have to show that $\lim_{N\to\infty} J_N^* = J^*$. If the method focuses on nested partitions, then many subsequences convergent to J^* will be found. For example, as depicted in Figure 1, for $N = 2, 4, 8, \cdots$, we have $P_2 \subseteq P_4 \subseteq P_8 \subseteq \cdots$. Indeed, if the number of time divisions is of the form 2^N, then the convergence of the results in the discrete step will be guaranteed by the following theorem.

Theorem 2. *The sequence $\{J_{2^N}^*\}$ converges to J^* as N tends to infinity.*

Proof: Because of nesting mechanism, $P_2 \subseteq P_4 \subseteq \cdots \subseteq P_{2N} \subseteq \cdots$, any admissible control vector $(u_1, u_2, \cdots, u_{2^N})$ is also admissible for the problem with 2^{N+1} points. Therefore, $J_{2^{N+1}}^* \leq J_{2^N}$, and the sequence is decreasing:

$$J_2^* \geq \ J_4^* \geq \cdots \geq J_{2^N}^* \geq \cdots \geq J^* \tag{17}$$

Since every decreasing and bounded sequence in real numbers converges to its infimum, then $\lim_{N\to\infty} J^*_{2^N} = J^o$. Now if by contradiction, $J^o < J^*$, then there exists an N such that $J^o < J^*_{2^N} < J^*$. On the other hand, there are a set of control like $(u_0, u_1, \cdots, u_{2^N})$ where, $J^*_{2^N} = J(u_0, u_1, \cdots, u_{2^N})$. Now, we construct a piecewise control function with this values as an input for the original continuous problem:

$$u(t) = \sum_{k=0}^{N} \chi_{[t_k,t_{k-1}]}(t) u_k, \tag{18}$$

where $\chi_{[t_k,t_{k-1}]}$ is the characteristic function on $[t_k, t_{k-1}]$ defined as:

$$\chi_{[t_k,t_{k-1}]}(t) = \begin{cases} 1 & \text{if } t \in [t_k, t_{k-1}] \\ 0 & \text{else} \end{cases} \tag{19}$$

If the above control function is feed to the system as input, then the response is $x(t)$ and the corresponding performance index $J(x, u)$ will be equal to $J^*_{2^N}$ which is less than J^*. This means that there are a pair of control-state with lower performance index than the optimal objective. This contradicts with the optimality of J^*. Therefore, $J^o = J^*$. $\square$

3. Proposed technique

As mentioned before, WOA is a population-based meta-heuristic optimization algorithm. Being competitive with other meta-heuristic optimizers, this technique is suitable for solving the discrete problem the pervious section. Due to its ease of implementation, proved accuracy, high rate of convergence, accepting bounds on decision variables, and applicability on large scale problem, WOA is applied to the problem of minimizing (8) in this section. The steps of the method are as follows:

Initialization: At the first step, a population of whales or search agents are selected. A set of control values is assigned randomly in the control interval $[l_b, u_b]$, such as $\{u_0, u_1, \cdots, u_N\}$. Let assume that $U^i(0)$ denotes values of the control function assigned to i-th whale at the initialization step.

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

Finding the best solution: Among whales there is one with the best related performance index. In order to find it, the corresponding states of all controls are calculated from (6), then the related performance index is determined by (8). By a simple comparing between whales' performances, the optimum will be found.

Update: The control values of each whale is 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:

$$U^i(t+1) = \begin{cases} U^*(t) - A.D, & \text{if } \ p < 0.5 \\ \\ D_i' d.e^{bl}.\cos(2\pi l) + U^*(t), & \text{if } \ p \geqslant 0.5 \end{cases} \tag{20}$$

where, $U^*(t)$ denotes the best solution at the t-th step. With this optimal solution, the other whales are updated in the next step. $D_i = |C.U^*(t) - U^i(t)|$ is the weighted distance between the i-th whale and the best solution sofar, $A = 2.a.r - r$ is a coefficient vector, $C = 2.r$ is a coefficient vector, a is a vector of constants linearly decreased from 2 to 0 during iterations, C is a random vector in $[0, 1]$, $D_i' = |U^*(t) - U^i(t)|$ is the distance between the i-th whale and the best solution sofar, b is a constant defining the shape of the logarithmic spiral, l is a random number in $[-1, 1]$, and . denotes element wise multiplication.

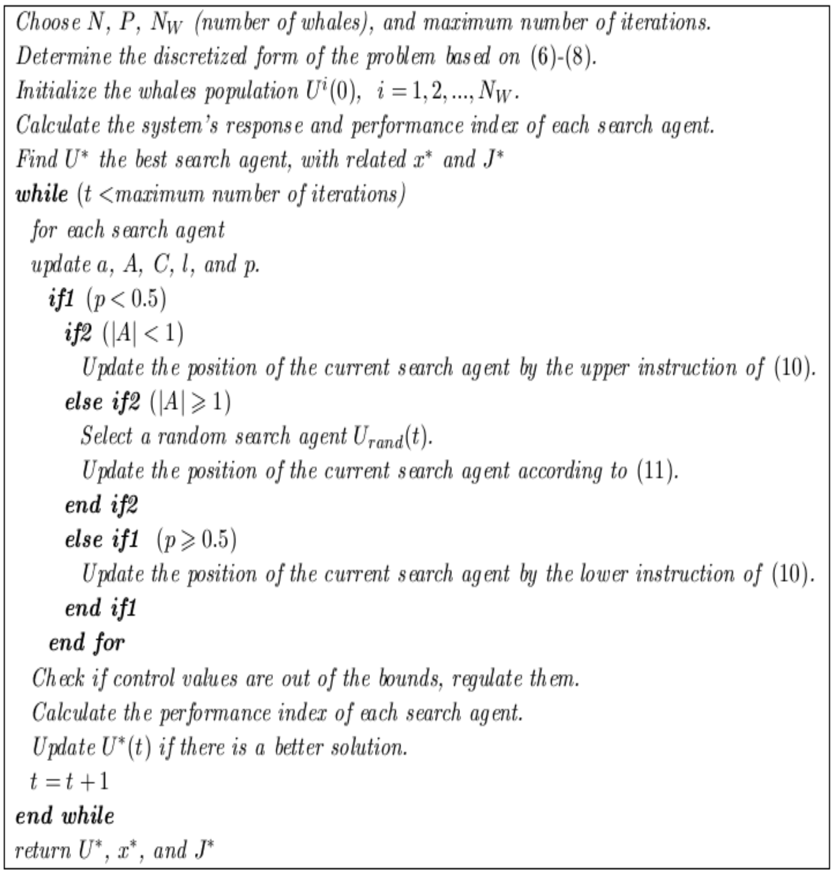
The equation of (20) is called the exploitation phase of updating. In the exploration phase, which performs when $p < 0.5$ and $|A| > 1$, the position of a search agent is updated based on a randomly chosen search agent instead of the best search agent found so far. This step bring out the WOA algorithm from local optimums and leads to a global search. The updating relation is as follows:

$$U^i(t+1) = U_{rand}(t) - A.D, \tag{21}$$

where, $D = |C.U_{rand}(t) - U^i(t)|$, and U_{rand} is a randomly admissible control. The main algorithm of WOA for optimal control problem is summarized in Figure 2.

3.1. Smoothing Technique

In the early stages of developing the proposed algorithm, we observed that the control function fluctuations, despite of controlling the system and minimizing the performance index. As this fluctuations and corners in the optimum solution make the implementation of it to a controller hard, usually an averaging method is applied in the post processing phase to smooth the solution. However, in the present method we tried to load smoothing process



```
Choose $N$, $P$, $N_W$ (number of whales), and maximum number of iterations.
Determine the discretized form of the problem based on (6)-(8).
Initialize the whales population $U^i(0)$,  $i = 1, 2, ..., N_W$.
Calculate the system's response and performance index of each search agent.
Find $U^*$ the best search agent, with related $x^*$ and $J^*$
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 (10).
      else if2 ($|A| \geqslant 1$)
        Select a random search agent $U_{rand}(t)$.
        Update the position of the current search agent according to (11).
      end if2
      else if1 ($p \geqslant 0.5$)
        Update the position of the current search agent by the lower instruction of (10).
      end if1
    end for
  Check if control values are out of the bounds, regulate them.
  Calculate the performance index of each search agent.
  Update $U^*(t)$ if there is a better solution.
  $t = t + 1$
end while
return $U^*$, $x^*$, and $J^*$
```

Figure 2: Whale optimization algorithm for optimal control problem.

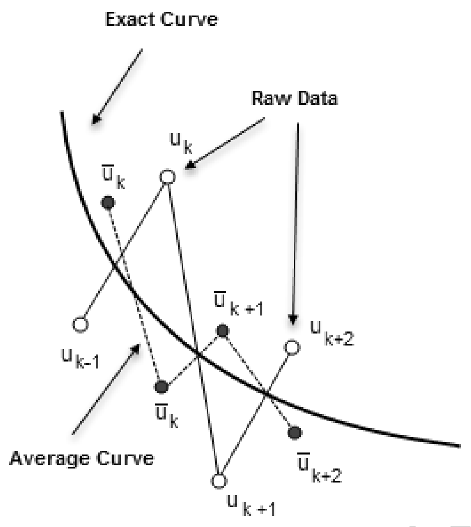


Figure 3: A schematic showing the effect of smoothing at tree typical points. White points are control values before smoothing and black points are corresponding smoothed ones.

during the iteration. To this end, smoothing is applied to the control values after updating and regulation phase.

The process of smoothing is done by an averaging based on the values of each tree successive points. The first step in smoothing is to identify the zigzag shapes in raw data. When the slope of two preceding segment is different in sign, that is the slope of the curve changes from positive to negative or vice versa, than a zigzag occurs as in Figure 3 between u_{k-1}, u_k, and u_{k+1}. In this case, the aim is to reduce the magnitude of jumps, which is performed by an averaging. If $0 < \alpha < 1$ is a weight coefficient, then the proposed formula for smoothing at point k is as follows:

$$\bar{u}_k = \begin{cases} \alpha u_{k-1} + (1-\alpha)u_{k+1}, & \text{if } \operatorname{sign}(u_k - u_{k-1})\operatorname{sign}(u_{k+1} - u_k) < 0 \\ u_k, & \text{else} \end{cases} \tag{22}$$

where, $\bar{u}_k$ is the smoothed value of u_k and replaced it during the next steps of iterations. The following theorem guarantees that the smoothing process does not violate control bounds.

Theorem 3. *Let $\{u_k, k = 0, 1, \cdots, N\}$ be an admissible control function in discrete form. Then its smoothed is also admissible.*

Proof: As the original control is admissible,

$$l_b \leqslant u_{k-1}, u_k, u_{k+1} \leqslant u_b$$

Now, if $\text{sign}(u_k - u_{k-1})\text{sign}(u_{k+1} - u_k) < 0$,

$$\bar{u}_k = \alpha u_{k-1} + (1-\alpha)u_{k+1} \leqslant \alpha u_b + (1-\alpha)u_b = u_b,$$

$$\bar{u}_k = \alpha u_{k-1} + (1-\alpha)u_{k+1} \geqslant \alpha l_b + (1-\alpha)l_b = l_b,$$

otherwise, $\bar{u}_k = u_k \in [l_b, u_b]$. Therefore, $\{\bar{u}_k, k = 0, 1, \cdots, N\}$ is also admissible.$\square$

It is expected that adding the smoothing routine to the algorithm, more smooth solution and closer to the exact one will be obtained. This claim is studied through the examples.
Noted that, this smoothing is recommended for problems with continuous optimal control. In the cases with switching control function, smoothing will highly affect the optimum performance by eliminating useful oscillations.

3.2. Parallel Processing Version

Parallel implementation will result in reduction of time and the ability of solving more complex problem. Population-based optimization algorithms can be parallelized because of including independent iterations over the huge number of data. There are many function evaluation which may performed simultaneously on different processors. One of the examples is the ant colony optimization method which is applied to solve optimal control problems in parallel in [11].
For the present method, there are some steps that could be performed in parallel to benefit from computer resources more.

In each iteration, there is a step of calculating performance index for all whales. This step is computationally expensive and time consuming due to solving the initial value problems by Runge-Kutta. Also, the process of updating the position of a whale is independent from others in WOA. Therefore, this step is a suitable section to be parallelized among computational resources.

Let consider a pool of n_p processors or threads in a shared memory computer. Then the total task including calculation of performance indices for

N_W agents is divided between n_p processor. Each processor calculates the performance index of its share and return all data for finding the best one to a master node. Therefore, it is expected that the running time of this step be reduces n_p times.
The flowchart in Figure 4 shows the improved algorithm with parallelization and with smoothing. The first steps of discretization and initialization run is a serial mode, then the execution enters the parallel region. After calculation of performance indices, running transfers to the serial mode again in order to find the optimum value among the whales. The controls are updated, regulated relative to upper and lower bounds, and smoothed based on the related scheme. Then the process iterates again until the maximum permitted number of iteration achieves.

3.3. Estimating the Computational Complexity

In order to determine the computational complexity of the proposed algorithm, one iteration is considered. Based on the flowchart of Figure 4, a typical iteration includes solving the system equations, finding performance indices, search for optimum, updating, regulation, and smoothing. Because of double for loops in last four subroutines, the upper bound of their computational complexity if of $O(NN_w)$, where N is the number of subintervals and N_w is the number of whales.

The phase of solving equations and performance indices calculation contains three nested for loops, due to multi-staging therefor, their computational complexity is of order $O(NN_wP)$, where, as mentioned before, P is the number of points in the partitioning the subintervals. Therefore, in the parallel version, the total computational complexity is $O(NN_wP)+O(NN_w)$. If the parallel processing is performed using n_p processors, the total complexity will be $O(\frac{NN_wP}{n_p}) + O(NN_w)$. Indeed, as indicated in flowchart, there are some parts running in serial. Now, the sequential fraction of the program is:

$$f_s = \frac{O(NN_w)}{O(NN_wP) + O(NN_w)} \approx \frac{1}{P+1} \tag{23}$$

Based on the Amdahl's law [20], the speedup of parallel program is dominated by the inverse of the sequential fraction, therefore, the maximum speed up is $(P+1)$. This says that using more that $(1+P)$ processor, there is no expectation of increment of speedup. This will be checked in numerical examples in the next section.

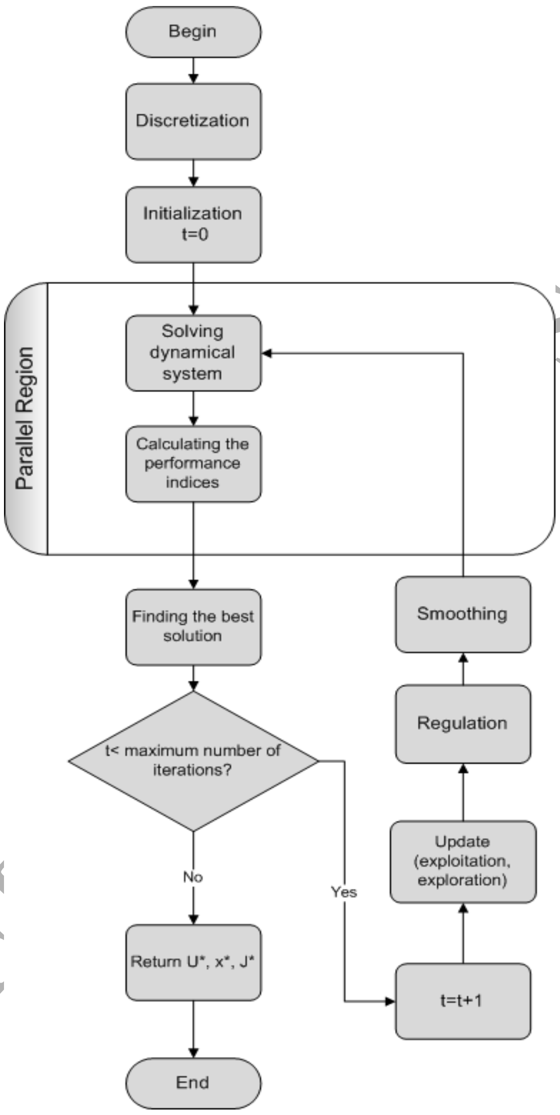

Begin
Discretization
Initialization t=0
Parallel Region
Solving dynamical system
Calculating the performance indices
Finding the best solution
t< maximum number of iterations?
No
Yes
Return U*, x*, J*
End
t=t+1
Update (exploitation, exploration)
Regulation
Smoothing

Figure 4: Flowchart of the algorithm.

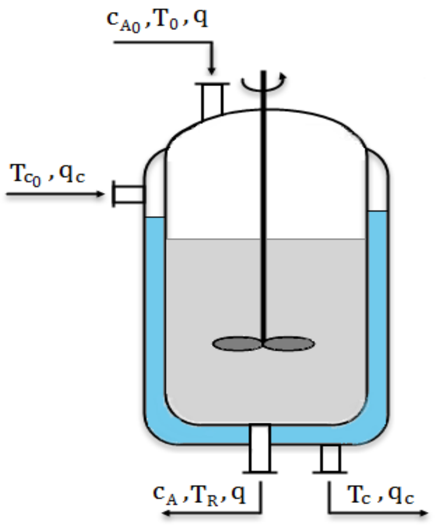


Figure 5: Schematic of Continuous Stirred Tank Reactor.

4. Numerical Examples

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

Example 1. Continuous Stirred Tank Reactor (CSTR)
The CSTR is a stirred vessel used to carry out the exothermic reactions. As depicted in Figure 5, there is a surrounding coolant liquid that transfers the heat of reaction out of the tank. The coolant flow rate is usually controlled by a solenoid valve. The problem of controlling coolant flow rate is a nonlinear control problem and has been studied in by researchers. For example, adaptive control of CSTR is presented in [23].
The governing equations for the reaction A → B in this CSTR which come from the heat and mass balances are as follows [9]:

$$Vc_P\rho\frac{dT_R}{d\theta} = qc_P\rho(T_0 - T_R) - VU^* - (-\Delta H)VR \tag{24}$$

$$V\frac{dc_A}{d\theta} = q(c_{A_0} - c_A) - VR \tag{25}$$

where, c_{A0}(kmol.m^{-3}) is the concentration of species A in feed stream, c_A(kmol.m^{-3}) is the concentration of species A leaving reactor, T_0(K) is the temperature

Table 1: Value of parameters in solving the problem of CSTR.

Number of whales	Maximum iterations	l_b	u_b	t_f	N	P	α
400	500	0.0	4.0	0.78	50	10	0.5

of feed stream to reactor, T_R(K) is the temperature of reactants leaving reactor, θ(min) is time, V(m^3) is the volume of reactor, q is the volumetric flow rate of feed and output streams, R(mol.m^{-3}.min^{-1}) is the reaction rate term, $-\Delta H$(kJ.mol^{-1}) is the heat of reaction, c_P(kJ.kg^{-1}.K^{-1}) is the heat capacity of reacting mixture, ρ(kg.m^{-3}) is density of reacting mixture, and U^*(kJ.m^{-3}.min^{-1}) is the rate of heat per unit volume removed from the reactor by the cooling system.

According to [9], (24)-(25) could be transformed into the following equations:

$$\frac{dx_1}{dt} = -(2+u)(x_1+0.25) + (x_2+0.5)\exp\left(\frac{25x_1}{x_1+2}\right) \tag{26}$$

$$\frac{dx_2}{dt} = 0.5 - x_2 - (x_2+0.5)\exp\left(\frac{25x_1}{x_1+2}\right) \tag{27}$$

where, x_1 is deviation from dimensionless steady-state temperature and x_2 is deviation from dimensionless steady-state concentration.
The aim of optimal control is to find a control function u that transfers the system from initial condition $(x_1(0), x_2(0)) = (0.09, 0.09)$ to the final desirable state $(x_1(t_f), x_2(t_f)) = (0, 0)$ while minimizes the following performance index:

$$J = \int_0^{t_f} \left(x_1^2 + x_2^2 + 0.1u^2\right) dt \tag{28}$$

where, t_f is the dimensionless final time.
This problem has an exact solution with $I^* = 0.133094$. The problem was solved with the proposed method of WOA with the parameter of Table 1. The best optimal objective was $I = 0.133724$ which is close to the exact one.

In Figure 6, the resulting control functions with and without smoothing are compared with the exact one. As expected, with the smoothing process

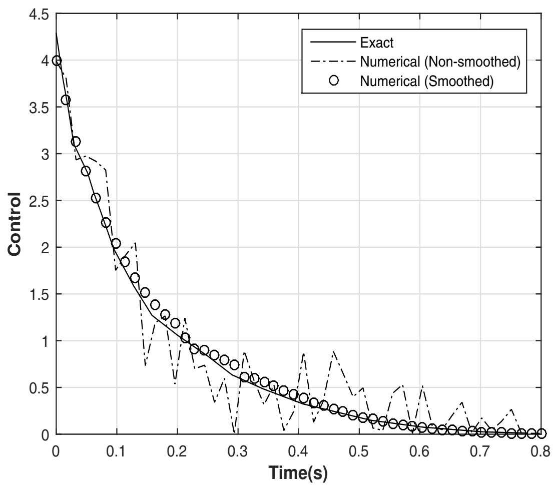


Figure 6: Comparison of numerical and exact solution of Example 1.

the control function is obtained more smooth and closer to the exact solution. Moreover, the performance index in terms of iteration is depicted in Figure 7 which shows the rate of convergence. As expected, when smoothing is applied, the required iterations number is decreased which serves the time and computational resources.

Then the parallel version was executed to check the amount of time reduction. It was tried on parallel computer of Aerospace Research Institute which has 48 1.8GHz-cores. The code was developed in C++ programming language using OpenMP application programming interface. It was compiled with g++ compiler version 4.9.2 and executed under Ubuntu 13 operating system.
Table 4 shows the wall clock time of running up to 16 threads. It is deduced from this table that by using parallel programming the time of the execution reduced from 33.24(sec) to 2.97(sec) if 16 threads are available. This leads to 69.9% parallel performance which shows the well working of parallelization. Figure 8 shows the wall clock time of running indicating the reduction rate in time of execution. The curve of speedup is also given in Figure 9 and compared with the ideal speedup. It is clear that the speedup is close to ideal case up to 8 threads and its divergence behavior after 8 threads is natural because of serial fraction of the algorithm and other overheads. As stated in Section 3.3, the speedup is restricted to $(P + 1)$. In this example, $P = 10$,

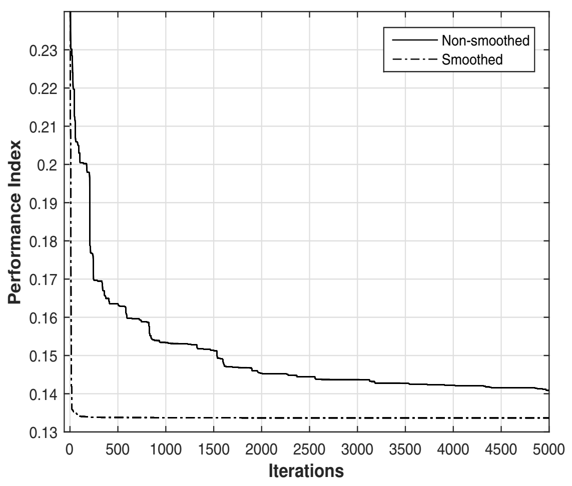


Figure 7: The performance index of Example 1 versus iterations number.

Table 2: Parallel performance results of solving Example 1.

Number of threads	Wall clock time (sec)	Speedup	Performance(%)
1	33.24	1	100.0
2	16.98	1.96	98.0
3	11.70	2.84	94.7
4	8.97	3.70	92.5
5	7.36	4.52	90.4
6	6.32	5.26	87.7
7	5.58	5.96	85.1
8	4.96	6.70	83.7
9	4.54	7.32	81.3
10	4.17	7.97	79.7
12	3.70	9.98	83.2
14	3.29	10.10	72.1
16	2.97	11.19	69.9

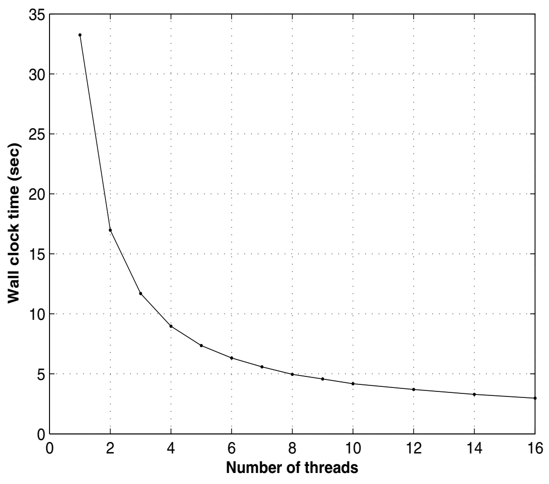


Figure 8: Wall clock time of executing Example 1.

therefore, no more speedup achievement is expected over 11 threads. This fact is also confirmed by Figure 9.

Example 2. Low-Thrust Rendezvous
Consider a spacecraft in a central gravity field. The equation of motion is expressed as:

$$\ddot{\mathbf{r}} = -(\mu/r^3)\mathbf{r} + \mathbf{F}/m \tag{29}$$

where, $\mathbf{r}$(km) is the position vector of the spacecraft from the center of gravity, $r = |\mathbf{r}|$, $\mu(\mathrm{km}^2\mathrm{s}^{-2})$ is the gravitational parameter of the central body, m(kg) is the constant mass of the spacecraft, and $\mathbf{F}$(N) is the non-gravitational control force vector.

By introducing a coordinate frame in [16], that is rotating along a circular orbit at a constant angular velocity, and assuming planar motion, the equation is reduced to

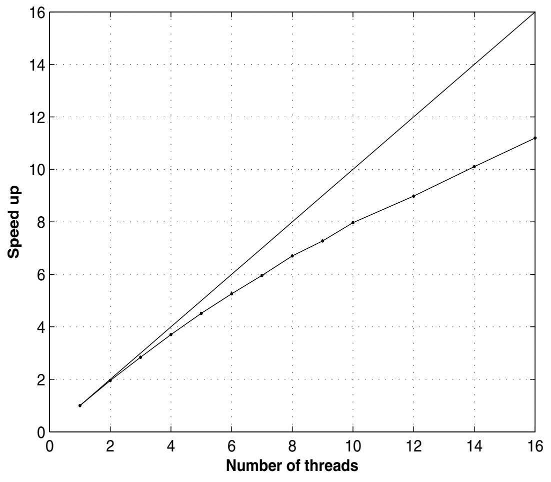


Figure 9: The curve of speedup for Example 1.

$$\dot{x_1} = x_3 \quad (30)$$
$$\dot{x_2} = x_4 \quad (31)$$
$$\dot{x_3} = 2x_4 - (1 + x_1)\left(\frac{1}{r^3} - 1\right) + u_1 \quad (32)$$
$$\dot{x_4} = -2x_3 - x_2\left(\frac{1}{r^3} - 1\right) + u_2 \quad (33)$$

Now, $r = \sqrt{(x_1 + 1)^2 + x_2^2}$, (u_1, u_2) is the thrust control vector with radial and tangential component. The state vector $x = (x_1, x_2, x_3, x_4)$ includes the radial and tangential displacements x_1, x_2 and the radial and tangential velocity deviations x_3, x_4. The aim is to control the states from initial point $x(0) = (0.2, 0.2, 0.1, 0.1)$ to $x(t_f) = (0, 0, 0, 0)$ in minimum control effort. Therefore, a combined performance index such as

$$J(x, u) = \frac{1}{2}x(t_f)^T S x(t_f) + \frac{1}{2}\int_0^{t_f}(u_1^2 + u_2^2)dt \quad (34)$$

has to be minimized, where S is a weighing diagonal matrix. This problem was solved in [21] by an iterative approximation method based on time-varying linear quadratic regulators. In [16], the linearized form of (33) was

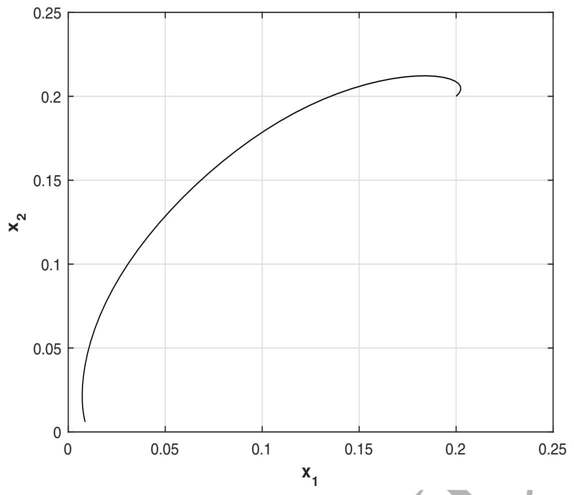


Figure 10: Numerical solution of Example 2 (x_2 versus of $x1$).

Table 3: Value of parameters in solving the problem of low thrust rendezvous.

Number of whales	Maximum iterations	l_{1b}	u_{1b}	l_{2b}	u_{2b}	t_f	N	P	α
400	150	-1.2	2.0	-2.5	0.4	1	50	10	0.5

used to find optimal feedback control based on first order necessary condition of optimality. A numerical method based on parallel ant colony optimization was also tested on this problem in [11].

The problem was solved with the proposed method on WOA with the parameters of Table 3. The best optimal objective was $I = 1.03020$ which is close to the result of [11] i.e. $I = 1.02$. The final conditions were obtained as $x(t_f) = (0.0059,\ 0.0087, -0.0230, 0.0033)$ that also satisfies the desired final condition.

Figure 10 shows the results of the method for x_1, x_2 states or responses to the obtained optimal control. Figure 11 is also depicted x_4 in terms of x_3 and indicates that the final velocities are zero as desired. In Figure 12 the obtained control component are given where, u_2 is plotted with respect of u_1.

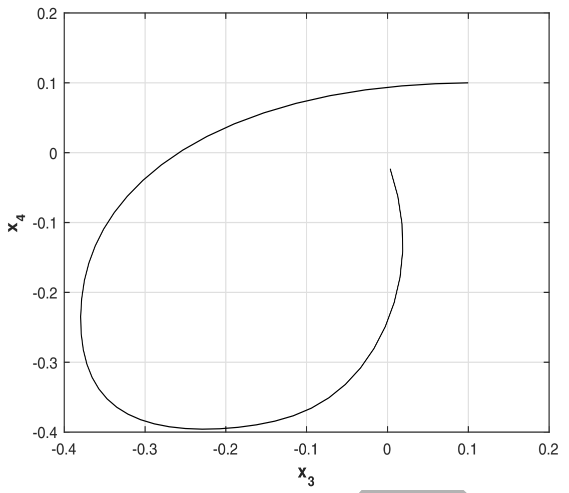


Figure 11: Numerical solution of Example 2 (x_4 versus of $x3$).

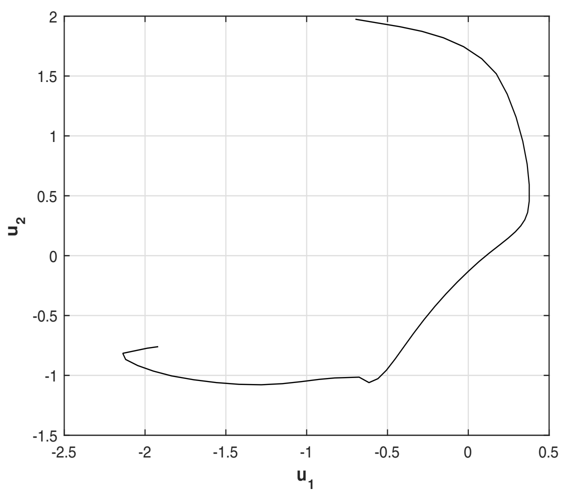


Figure 12: Numerical solution of Example 2 (u_2 versus of $u1$).

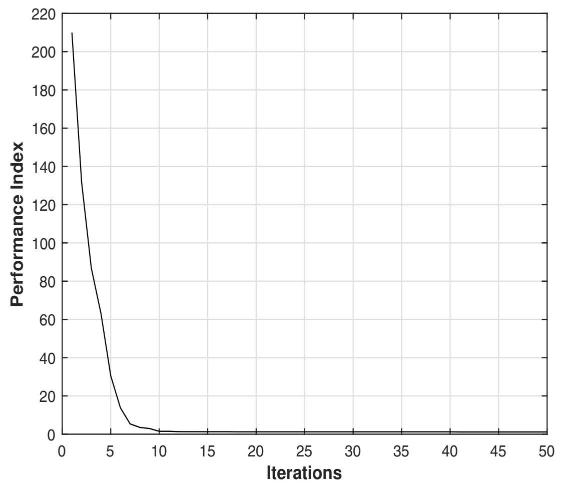


Figure 13: The performance index of Example 2 versus iterations number.

The problem was solved also in parallel format on the parallel computer of Aerospace Research Institute. The parallel form works as well as the serial one with similar results. The speedup curves are shown in Figure14. It is deduced from these curves that the speedup has acceptable linear trend up to 16 threads and after that reaches to a margin because of serial fraction of the algorithm and other overheads. It is also close to the ideal speedup curve up to 10 threads. Therefore, for this example, using up to 16 threads is recommended.
In order to check the effect of whales' number (or search agents) on speedup, the program was examined with 200, 400, and 800 whales. As is distinguished form Figure14, there are not meaningful differences with them. Therefore, as expected from Section 3.3, the number of whales will only affect the accuracy of the solutions and wall clock time of executing the program.
One of the other parameters is N, the number of subintervals in discretization phase. In order to study this effect, the problem is solved for $N = 50$, $N = 100$, and $N = 200$, while $P = 50$ is fix. It results that the time of execution is increased linearly with increase of N, while the speedup has no considerable difference as plotted in Figure 15. This also confirms the result of Section 3.3, where it was proved that speed up is dependent from N and N_W.

Example 3. Minimum Energy

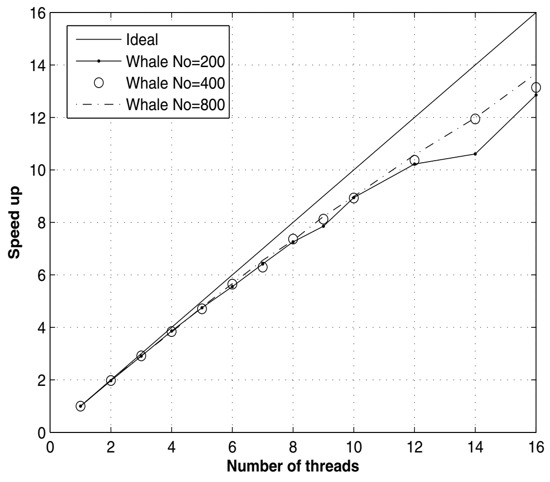


Figure 14: The speedup curves for different number of whales.

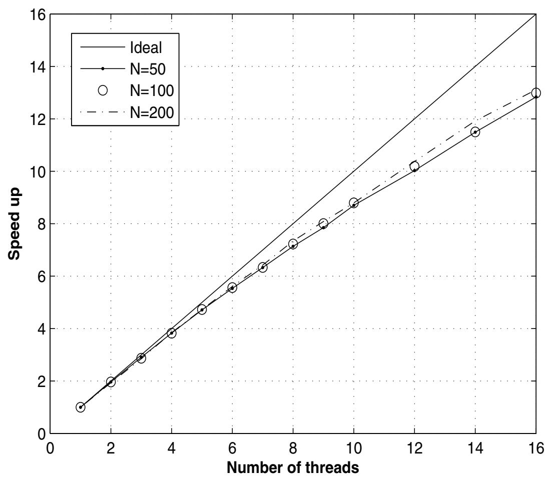


Figure 15: The speedup curves for different number of subintervals in discretization phase.

Table 4: Value of parameters in solving Example 3.

Number of whales	Maximum iterations	l_b	u_b	t_f	N	P	α
400	500	-18.0	18.0	1.0	35	10	0.5

As the last example a minimum energy problem will be studied. Let consider the following linear dynamical system [2]:

$$\dot{x}_1(t) = \dot{x}_2(t) \tag{35}$$
$$\dot{x}_2(t) = -x_2(t) + u(t) \tag{36}$$

The aim of minimum energy problem is to control the system form an nominal starting point, for example $(1, 1)$ to a final destination at origin in such a way the the following performance index is minimized:

$$J = \frac{1}{2}\int_0^{t_f} u^2(t)dt \tag{37}$$

where $t_f = 1$. The problem has analytical solution obtained by Pontryagin maximum principle [1] as

$$x_1(t) = -\frac{1}{2}C_1e^t + C_2e^{-t} + C_4 \tag{38}$$
$$x_2(t) = -\frac{1}{2}C_1e^t + C_2e^{-t} + C_3 + C_4t \tag{39}$$

where,

$$C_1 = \frac{4e-6}{e^2-4e+3}, C_2 = \frac{e^2}{e^2-4e+3}, C_3 = \frac{-2e}{e^2-4e+3}, C_4 = \frac{2e}{e-3} \tag{40}$$

This example was chosen because of exciting exact solution and results of some other meta-heuristic methods in order to make a comparison between the proposed method and pervious works. Therefore, the problem was solved with method parameters as given in Table 4.

The results of system' responses are also depicted in Figure 16 and Figure 17 and compared with the exact solution (38)-(40), and results of GA, IWO and PSO which have been taken from [2]. It is clear that that the curve

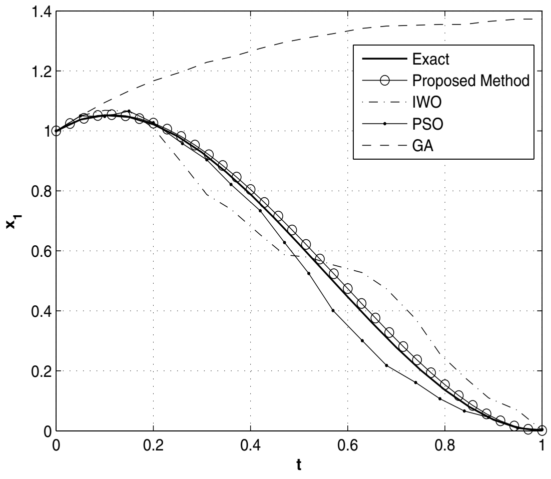


Figure 16: The exact and numerical responses of first channel in Example 3.

of applying the proposed method have the best fitting with the exact solution with respect to above mentioned numerical methods. This is because of continuous search of WOA for control values and smoothing process. The responses of the system to the optimal control satisfies the final conditions with a miss distance of order 10^{-3}.

5. Conclusion

A numerical method for optimal control problem was presented. The method utilizes the whale optimization algorithm as the core of optimization part. The method was also improved by smoothing and implementing parallel computing features. Results of implementing the proposed method on case studies show that the method is accurate for nonlinear problems. Comparing the results of the method with existing meta-heuristics for optimal control problems reveals that the proposed method leads to solutions which are more smooth and more closer to the exact solutions.

Examining parallel running of the method shows that the method benefits from parallelism which decreases the time of execution.

For future researches, the methods can be extended to solve minimum time optimal control problem, where the end of time interval is not fix. Optimal control problems on distributed dynamical systems may be analyzed

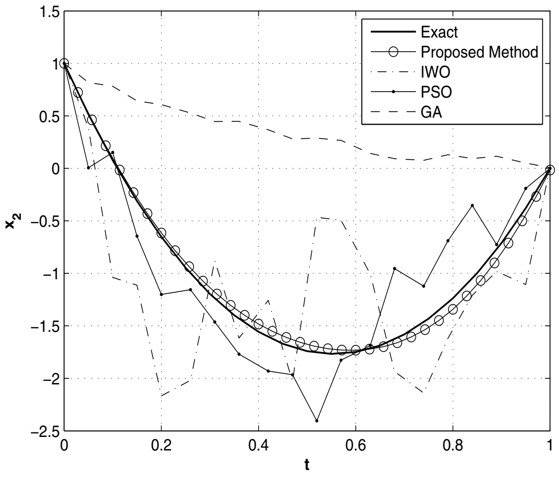


Figure 17: The exact and numerical responses of second channel in Example 3.

with WOA as well. From computational aspects the method can be adopted on computers with GPU in order to benefit form graphical processing acceleration and to compare the parallel performance on shared memory structures.

References

[1] L.T. Aschepkov, D.V. Dolgy, T. Kim, R.P. Agarwal,it Optimal Control, Springer International Publishing, 2016.

[2] A.H. Borzabadi, M. Heidari, Comparison of Some Evolutionary Algorithms for Approximate Solutions of Optimal Control Problems, *Australian Journal of Basic and Applied Sciences*, vol.4, no.8, pp.3366-3382, 2010.

[3] A. H. Borzabadi, H. H. Mehne, Ant Colony Optimization for Optimal Control Problems, *Journal of Information and Computing Science*, vol.4, no.4, pp.259–264, 2009.

[4] S. Boubaker, F. M'sahli, Solutions Based on Particle Swarm Optimization for Optimal Control Problems of Hybrid Autonomous Switched Sys-

tems,*International Journal of Intelligent Control and Systems*, vol. 13, no.2, pp.128–135, 2008.

[5] S. Boubaker, M. Djemai, N. Manamanni, F. Msahli, O ptimal Control of Switched Linear Systems by Particle Swarm Optimization, *IFAC Proceedings Volumes*, vol. 42, no. 19, pp.484-489, 2009.

[6] A. Ghosh, S. Das, A. Chowdhury, R. Giri, An ecologically inspired direct search method for solving optimal control problems with Bzier parameterization, *Engineering Applications of Artificial Intelligence*, vol.24, no.7, pp. 1195-1203, 2011.

[7] Q. Lin, R. Loxton, K.L. Teo, The Control Parameterization Method for Nonlinear Optimal Control: a survey, *Journal of Industrial and Management Optimization*, vol.10, no.1, pp.275–309, 2014.

[8] B. Luo, H.N. Wu, T. Huang, D. Liu, Data-based approximate policy iteration for affine nonlinear continuous-time optimal control design, *Automatica*, vol.50, no.12, pp.3281-3290, 2014.

[9] R. Luss, *Iterative Dynamic Programming*, Chapman & Hall, CRC, 2000.

[10] H.H. Mehne, and A.H. Borzabadi, A numerical method for solving optimal control problems using state parametrization, *Numerical Algorithms*, vol.42, no.2, pp.165-169, 2006.

[11] H.H. Mehne, Evaluation of Parallelism in ACO Method for Numerical Solution of Optimal Control Problems, *Journal of Electrical Engineering, Electronics, Control and Computer Science*, vol.1, no.2, pp.15-20, 2015.

[12] Z. Michalewicz, J.B. Krawczyk, M. Kazemi, C.Z. Janikow, Genetic algorithms and optimal control problems, *Proceedings of the 29th IEEE Conference on Decision and Control*, pp.1664–1666, 1990.

[13] Z. Michalewicz, C. Z. Janikow, J. B. Krawczyk, A modified genetic algorithm for optimal control problems, *Computers and Mathematics with Applications*, vol. 23, no. 12, pp. 83–94, 1992.

[14] S. Mirjalili, A. Lewis, The Whale Optimization Algorithm, *Advances in Engineering Software*, vol.95, pp.51-67, 2016.

[15] S. Nezhadhosein, A. Heydari, R. Ghanbari, A Modified Hybrid Genetic Algorithm for Solving Nonlinear Optimal Control Problems, *Mathematical Problems in Engineering*, vol. 2015, Article ID 139036, pp.1–21, 2015.

[16] C. Park, V. Guibout, D.J. Scheeres, Solving Optimal Continuous Thrust Rendezvous Problems with Generating Functions, *Journal of Guidance, Control, and Dynamics*, vol.29, No.2, 2006.

[17] A. Quarteroni, R. Sacco, F. Saleri, *Numerical Mathematics*, Springer, 2007.

[18] H. Seywald, R. R. Kumar, S. M. Deshpande, Genetic Algorithm Approach for Optimal Control Problems with Linearly Appearing Controls, *Journal of Guidance Control, and Dynamics* , vol. 18, no. 1, pp. 177–182, 1995.

[19] Y. Shimizu, An Evolutionary Approach to Derive Adaptive Optimal Control Policy for Chemical Reaction Processes, *10th International Symposium on Process Systems Engineering - PSE2009*, pp. 1185–1190, 2009.

[20] M. O. Tokhi, H. H. Hossain, M. H. Shaheed, *Parallel Computing for Real-time Signal Processing and Control*, Springer, pp.87-88, 2003.

[21] F. Topputo, F. Bernelli-Zazzera, Approximate Solutions to Nonlinear Optimal Control Problems in Astrodynamics, *ISRN Aerospace Engineering*, vol.2013, Article ID 950912, pp. 1-7, 2013.

[22] J.M. van Ast, R. Babuka, B. De Schutter, Ant Colony Learning Algorithm for Optimal Control. In: Babuka R., Groen F.C.A. (eds) Interactive Collaborative Information Systems. *Studies in Computational Intelligence*, vol 281. Springer, Berlin, Heidelberg, 2010.

[23] J. Vojtesek, P. Dostal, Nonlinear versus Ordinary Adaptive Control of Continuous Stirred-Tank Reactor, *The Scientific World Journal*, vol.2015, Article ID 389273, pp.1-10, 2015.

[24] Y. Yamashita, M. Shima, Computation Method for Optimal Control Problem with Terminal Constraints using Genetic Algorithm, *Transaction of the Society of Instrument and Control Engineers*, vol.E–1, no.1, pp. 274–280, 2001.