Engineering Applications of Artificial Intelligence 24 (2011) 476–484

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Solving nonlinear optimal control problems using a hybrid IPSO–SQP algorithm

Hamidreza Modares *, Mohammad-Bagher Naghibi Sistani

Department of Electrical Engineering, Ferdowsi University of Mashhad, Mashhad 91775-1111, Iran

ARTICLE INFO

Article history:
Received 24 April 2010
Received in revised form
30 July 2010
Accepted 3 August 2010
Available online 21 August 2010

Keywords:
Optimal control
Optimization
Particle swarm optimization
Inertia weight
Successive quadratic programming

ABSTRACT

A hybrid algorithm by integrating an improved particle swarm optimization (IPSO) with successive quadratic programming (SQP), namely IPSO–SQP, is proposed for solving nonlinear optimal control problems. The particle swarm optimization (PSO) is showed to converge rapidly to a near optimum solution, but the search process will become very slow around global optimum. On the contrary, the ability of SQP is weak to escape local optimum but can achieve faster convergent speed around global optimum and the convergent accuracy can be higher. Hence, in the proposed method, at the beginning stage of search process, a PSO algorithm is employed to find a near optimum solution. In this case, an improved PSO (IPSO) algorithm is used to enhance global search ability and convergence speed of algorithm. When the change in fitness value is smaller than a predefined value, the searching process is switched to SQP to accelerate the search process and find an accurate solution. In this way, this hybrid algorithm may find an optimum solution more accurately. To validate the performance of the proposed IPSO–SQP approach, it is evaluated on two optimal control problems. Results show that the performance of the proposed algorithm is satisfactory.

© 2010 Elsevier Ltd. All rights reserved.

1. Introduction

One of the most active subjects in control engineering is optimal control of nonlinear systems. Such control problems arise in many applications, e.g., in economics, chemical engineering, robotics and aeronautics. To solve optimal control for a nonlinear system using classic optimal control theory, the nonlinear Hamilton–Jacobi–Bellman (HJB) partial differential equations must be solved (Bryson and Ho, 1975). However, in practice, the nonlinear HJB equations are very difficult to be solved. As a result, it is necessary to employ numerical methods to solve nonlinear optimal control (NOC) problems.

Several works in the literature were proposed to solve NOC problems, numerically. Gradient descents are the most elegant and precise numerical methods to solve NOC problems. Nevertheless, they have the possibility of getting trapped at local optimum depending on the initial guess of solution. In order to achieve a good final result, these methods require very good initial guesses for control variable trajectory. Besides, as the complexity of the system increases, the specification of a suitable initial guess can become troublesome (Bayón et al., 2009). Alternatively, Iterative Dynamic Programming (IDP) (Luss, 2000) is a powerful method for solving optimization problems, but usually the CPU time used to solve the problem is quite long and it may also converge to the local optimum (Lopez Cruz et al., 2003; Bayón et al., 2009).

Since solving optimal control of nonlinear complex dynamics lead to presenting multiple local optimums, global optimal control approaches can be used to find the global optimum or a sufficiently close approximation. Heuristic optimization algorithms such as genetic algorithms (GA) (Holland, 1975; Goldberg, 1989); differential evolution (DE) (Price and Storn, 1997) and particle swarm optimization (PSO) (Kennedy and Eberhart, 1995) are found to have a better ability to converge to a global solution than the traditional methods in complex optimization problems (Onwubolu and Babu, 2004). Among their advantages are: (1) the objective function's gradient is not required; (2) they are not sensitive to initial guess of solution and (3) they usually do not get stuck into a local optimum. Based on these advantages, they have been successfully applied in many NOC problems (Sewald and Kumar, 1995; Yamashita and Shima, 1997; Lopez Cruz et al., 2003; Sarkar and Modak, 2004; Babu and Angira, 2006; Varadarajan and Swarup, 2008; Arumugam and Rao, 2008; Herrera and Zhang, 2009).

Recently, PSO algorithm has been found to be a promising technique for real world optimization problems (Clerc and Kennedy, 2002). Compared to GA, PSO takes less time for each function evaluation as it does not use many of GA operators like mutation, crossover and selection operator. Although PSO has shown some advances by providing high speed of convergence in specific problems, however, it does exhibit some shortages. First, it may convergence to a local optimum when facing with complex

* Corresponding author.
E-mail addresses: reza_modares@yahoo.com, Ha.modarres@stu-mail.um.ac.ir (H. Modares).

0952-1976/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.engappai.2010.08.002

optimization problems. Second, the convergence rate decreased considerably in the later period of evolution; when reaching a near optimal solution, the algorithm stops optimizing, and thus the achieved accuracy of algorithm is limited (Kennedy et al., 2001).

In this paper, a novel algorithm, namely IPSO–SQP, is proposed to overcome these two shortages. First, an improved PSO (IPSO) algorithm is proposed to enhance global search ability and also convergence speed of PSO. Second, to achieve faster convergence speed around global optimum and also higher convergence accuracy, the proposed IPSO is combined with successive quadratic programming (SQP) algorithm. Although the ability of SQP is weak to escape local optimum, it can achieve faster convergence speed around global optimum and the accuracy can be higher. The fundamental idea in the proposed method is that at the beginning stage of searching for the optimum, IPSO algorithm is employed to find a near optimum solution and accelerate the training speed. When the change in fitness value is smaller than a predefined value, or the particles in swarm being close to the global optimum, the best solution found by IPSO algorithm will be taken as the initial starting point for the SQP and the searching process is switched to SQP searching to accelerate the search process and find an accurate solution. In this way, this hybrid algorithm may find an optimum solution more quickly and accurately.

To our knowledge, the first and only work using a hybrid PSO–SQP method for an optimization problem was done by Victoire and Jeyakumar (2004). They used it for solving economic dispatch with valve-point effect. However, our proposed method has two distinctions with the previous one. First, an improved PSO is proposed to enhance global search ability of PSO algorithm and also increase the convergence speed and accuracy of PSO. Second this is the first research that used a hybrid PSO–SQP method for solving a dynamic optimization problem, i.e., nonlinear optimal control problems. In order to show the feasibility of the proposed method, two benchmark NOC problems are considered. The proposed algorithm is evaluated on these two NOC systems and its results are compared with those obtained by two well-known evolutionary algorithms, namely GA and DE algorithms, and also with three improved PSO algorithms. The results show that the proposed algorithm has better performance than others in terms of robustness and accuracy.

The rest of the paper is organized as follows: the next section describes a general form of optimal control problems. In Section 3, both GA and the DE algorithms are described. Section 4 introduces PSO algorithm. In Section 5, a brief description of the SQP algorithm for the solution of optimal control problems is given. Section 6 introduced the hybrid IPSO–SQP algorithm. Section 7 contains simulation results obtained by the proposed method when applying it to three benchmark optimal control problems. Finally, conclusion is presented in Section 8.

2. Optimization problem formulation

Consider a system described by the following nonlinear differential equation:

$$\dot{x} = f(x(t),u(t),t) \tag{1}$$

where $x(t) \in R^n$ is the state vector and the initial state $x(0)$ is given. $u(t) \in R^m$ is the control vector bounded by

$$u_{\min} < u_i(t) < u_{\max}, \quad i = 1,2,\ldots,m \tag{2}$$

Furthermore, it may be inequality constrains on state variables like

$$\psi_i(\mathbf{X}) \le 0, \quad i = 1,2,\ldots,l \tag{3}$$

The performance index associated with this system is a scalar function, which can be formulated as follows:

$$J(u(t)) = \phi(x(t_f),t_f) + \int_0^{t_f} L(x(t),u(t),t)\,dt \tag{4}$$

where J is a scalar performance index (PI), $\phi(.)$is final state cost function and $L(.)$ is interval cost function. $\phi(.)$and $L(.)$ functions are chosen to achieve an appropriate design goal. The objective is to determine the optimal control policy $u(t)$ in the time interval $t \in [t_0, t_f]$ such that PI is minimized or maximized.

To solve this type of problems numerically, there are two general approaches: indirect and direct methods (Stryk and Bulirsch, 1992). Indirect method is based on the solution of a calculus of variations problem through the use of the Pontryagin's minimum principle (PMP) (Bryson, 1999). In a direct approach, the optimal control problem is approximated by a finite dimensional optimization problem, which can be cast in a nonlinear programming (NLP) form and solved accordingly (Agrawal and Fabien, 1999). This is achieved through control parameterization. In our case the control $u(t)$ is assumed to be a piecewise constant such as (Lopez Cruz et al., 2003)

$$u(t_k) = u_k, \quad t \in [t_k,t_{k+1}], \quad k = 0,1,\ldots,N-1, \quad t_0 = 0, \quad t_N = t_f \tag{5}$$

As a result $N \times m$ parameters determine the control over $[0,t_f]$. The NLP problem is to find the stacked control vector defined by $\tilde{u} = [u_0^T,u_1^T,\ldots,u_{N-1}^T] = [\tilde{u}_1,\ldots,\tilde{u}_{N\times m}]$, where $\tilde{u}_i$, $i = 1,2,\ldots,m \times N$ are scalars.

3. Two classes of evolutionary algorithms: GA and DE

In this section, the two well-known evolutionary algorithms, namely GA and DE, which have been used earlier for solving NOC problems, are introduced briefly.

3.1. Genetic algorithm (GA)

Genetic algorithm (GA) (Holland, 1975; Goldberg, 1989) is a population based optimization technique that searches the best solution of a given problem based on the concepts of natural selection, genetics and evolution. The search is made starting from an initial population of individuals, often randomly generated. An individual is considered to be a possible candidate solution for the optimization problem in hand. At each evolutionary step, individuals are evaluated using a fitness function. The evolution (i.e., the generation of a new population) is made by means of three kinds of operator: breeding, mutation and selection. Selection involves killing a given proportion of the population based on probabilistic "survival of the fittest". Killed individuals are replaced by children, which are created by breeding the remaining individuals in the population. For each child produced, breeding first requires probabilistic selection of two parent individuals, getting a more chance to choose fitter individuals. Mutation allows new areas of the response surface to be explored by random alterations of optimization variables. GA iteratively improved the set of tentative solutions by applying the aforementioned stages to find a good solution.

In the traditional GA, all the variables of interest must be encoded as binary digits (genes) forming a string (chromosome). After a manipulation of binary-coded GA, the final binary digits are then decoded as original real numbers. On the other hand, in a real-coded GA, all genes in a chromosome are real numbers. To deal with practical engineering problems, the real-coded GA is more suitable than the binary-code GA (Chambers, 1995; Chang, 2007).

3.2. Differential evolution (DE)

As other evolutionary algorithms, the differential evolution (DE) proposed by Price and Storn (1997) is a population based optimization algorithm. DE starts with the random initialization of a population of individuals in the search space. Then, evolution mechanism is applied to parameter vectors. To the next iteration, only those parameter vectors of the procedure survive that produce the best performance for an objective function. The evolution mechanism contains a mutation procedure, which consists of adding a weighted difference between two vectors while creating new one and comparing a newly created vector to the one from an existing population. In order to increase the diversity of the muted vectors, one introduces the crossover. Crossover procedure generates trial vector by randomly mixing the muted vector with the last target vector. If the trial vector produces a better performance for an objective function than that compared to, the new vector replaces it in the population. This procedure is called selection. DE iteratively improved the set of tentative solutions by applying the aforementioned stages until satisfactory results are obtained or certain criteria of termination are met. We recommend the readers to refer to Price and Storn (1997) and Lopez Cruz et al. (2003) for further details.

4. Particle swarm optimization (PSO)

4.1. Standard PSO

Particle swarm optimization (PSO) is a heuristic population based optimization algorithm simulating the movement and flocking of birds (Kennedy and Eberhart, 1995). In the beginning of search process, a population of candidate solutions, called particles, is created randomly in the solution space. Each particle is associated with a velocity. Then, the velocity of every particle is iteratively adjusted according to the corresponding particle's experience and the particle's companions' experiences. It is expected that the particles will move towards better solution areas. The fitness of every particle can be evaluated according to the objective function of optimization problem. At each iteration, the velocity of every particle will be calculated as follows:

$$v_i^{t+1} = \omega v_i^t + c_1 r_1 (pbest_i^t - x_i^t) + c_2 r_2 (gbest^t - x_i^t) \quad (6)$$

where x_i^t is the position of the particle i in tth iteration,$pbest_i^t$is the best previous position of this particle (i.e., personal best), $gbest^t$ is the best previous position among all the particles in tth iteration (i.e., global best), ω is the inertia weight, c_1 and c_2 are acceleration coefficients and are known as the cognitive and social parameters, respectively. Finally, r_1 and r_2 are two random numbers in the range [0, 1]. After calculating the velocity, the new position of every particle can be worked out:

$$x_i^{t+1} = x_i^t + v_i^{t+1} \quad (7)$$

The PSO algorithm performs repeated applications of the update equations provided until a stopping criterion is met.

4.2. Inertia weight adaptation and the proposed PSO algorithm

The success of PSO during search is highly dependent on a good balance between exploration and exploitation. Exploration allows searching the entire search space by ensuring the redirection of the search towards new regions, while exploitation favors a quick convergence towards the optimum. To do a good balance between exploration and exploitation, one can use an appropriate adaptation mechanism for inertia weight factor. A big inertia weight facilitates exploration, but it makes the particle take long time to converge. Conversely, a small inertia weight facilitates exploration and makes the particle to converge fast, but it sometimes leads to local optimal.

Several researchers proposed PSO algorithms with an adaptive inertia weight (Shi and Eberhart, 1998; Chatterjee and Siarry, 2006; Modares et al., 2010a, 2010b). First of all, a PSO with linearly decreasing inertia weight (PSO–LDW) is proposed by Shi and Eberhart (1998). In PSO–LDW, the inertia weight linearly decreases as follows:

$$\omega^t = \omega_{\min} + \frac{iter_{\max} - t}{iter_{\max}} .(\omega_{\max} - \omega_{\min}) \quad (8)$$

where $iter_{\max}$ is the maximal number of iterations and t is the current number of iterations. So as iterations go, ω decreases linearly from $\omega_{\max}$ to $\omega_{\min}$.

After that several nonlinear inertia weight adaptation mechanisms were proposed to enhance the performance of PSO algorithm. Among them, Chatterjee and Siarry (2006) proposed the well-known PSO with nonlinearly decreasing inertia weight (PSO–NDW). The inertia weight starts with a high value $\omega_{\max}$ and nonlinearly decreases to $\omega_{\min}$ at the maximal number of iterations. This means that the representations are the same as those in the PSO–LDW method except that the inertia weight factor changes according to

$$\omega^t = \omega_{\min} + \left(\frac{iter_{\max} - iter}{iter_{\max}}\right)^{\alpha} (\omega_{\max} - \omega_{\min}) \quad (9)$$

As for $\alpha = 1$, the system becomes a special case of the method in Shi and Eberhart (1998).

Since the search process of PSO is nonlinear and highly complicated, linearly and nonlinearly decreasing inertia weight with no feedback taken from the global optimum fitness cannot truly reflect the actual search process. In the beginning of the search process, the particles are far away from the optimum point and hence a big inertia weight is needed to globally search the solution space. Conversely, when the best solution found by the population improves greatly after some iteration, i.e., the particles find a near optimum solution, only small movements are needed and inertia weight must be set to small values. Based on this, in one of our previous works, we proposed an improved PSO algorithm (Modares et al., 2010a) in which the inertia weight was set as a function of global optimum fitness during search process of PSO algorithm as follows:

$$\omega^t = 1/(1 + \exp(-\alpha \times F(gbest^t))) \quad (10)$$

where $F(gbest^t)$is the fitness of global best in tth iteration. The parameter α needs to be predefined. It can be set to the inverse of the value of global optimum fitness in the first iteration ($\alpha = 1/F(gbest^1)$). In this case, ω changes according to the rate of global best fitness improvement.

However, introducing the same inertia weight for all particles, by ignoring the differences among particle performances, simulated a roughly animal background, not a more precise biological model. In fact, during the search every particle dynamically changes its position, so every particle is located in a complex environment and faces a different situation. Therefore, every particle may have different tradeoffs between global and local search abilities.

Motivated by the aforementioned, to incorporate the difference between particles in PSO, similar to our recent work (Modares et al., 2010b), in this paper we developed an improved PSO (IPSO) in which the value of inertia weight for every particle in tth iteration is dynamically calculated by

$$\omega_i^t = 1/(1 + \exp(-\alpha F(pbest_i^t))) \quad (11)$$

This adaptation mechanism is like that described in Eq. (10), except replacing the global best fitness by the fitness of personal best fitness. Under the assumption above, it can be concluded that $0.5 \leq \omega_i < 1$.

According to Eq. (11), during the search of IPSO algorithm, while the fitness of a particle is far away from the real global optimal, the value of inertia weight will be large resulting in strong global search abilities and locating the promising search areas. Meanwhile, when the fitness of a particle is achieved near the real global optimal, the inertia weight will be set small, depending on the nearness of its best fitness to the optimal value, to facilitate a finer local explorations and hence accelerate convergence.

4.3. Performance analysis of the proposed PSO algorithm

In order to analyze the performance of the proposed IPSO, two constraint benchmark optimization problems (Mathur et al., 2000) are considered and listed in Table 1. Originally PSO algorithms are designed to solve unconstrained static optimization problems. To our knowledge, the penalty function method has been the most popular constraint-handling technique due to its simple principle and ease of implementation (Homaifar et al., 1994; Joines and Houck, 1994; Coello, 2000). The violations of constraints of the solutions are incorporated into the objective function so that the original constrained problems are transformed into unconstrained ones. Thus, in this method, the fitness function is defined as the sum of the objective function and a penalty term that depends on the constraint violation.

To compare the accuracy of IPSO with other aforementioned PSO algorithms, a maximum iteration of 100 is considered as a stopping condition. In addition, in all PSO–LDW, PSO–NDW and IPSO algorithms, we set $c_1=c_2=2$ as suggested by Shi and Eberhart (1998). In PSO–LDW and PSO–NDW, ω decreases from 0.9 to 0.4. Moreover, in NDW–PSO n is set to 1.2 (Chatterjee and Siarry, 2006).

Tables 2 and 3 list the results obtained by each algorithm, where each algorithm is implemented 20 times independently, for a population size of 20 and 40, respectively. As shown in Tables 2 and 3, it is clear that the worst result obtained by IPSO is similar to or even better than the best result obtained by others. Also, Figs. 1 and 2 show how PSO algorithms convergence to the global optimum for functions 1 and 2, respectively. It is clearly obvious that the proposed IPSO has a great advantage of convergence speed compared to PSO–LDW and PSO–NDW algorithms.

5. SQP algorithm

SQP is a nonlinear programming method that starts from a single searching point and finds a solution using the gradient information. Although this optimizing method is less time consuming than the population based search algorithms, it is highly dependent on the initial estimate of solution (Costa et al., 2005; Bayón et al., 2009).

The method resembles closely to Newton's method for constrained optimization just as is done for unconstrained optimization. SQP is based on iterative formulation and on the solution of quadratic programming sub-problems. The sub-problem is obtained

Table 1
Constrained benchmark optimization problems used for comparison of PSO algorithms.

	Test functions	Optimal value	Lower limits	Upper limits
f_1	$\max F = x_1^2+x_2^2+x_3^2$ s.t. $4(x_1-0.5)^2+2(x_2-0.2)^2+x_3^2+0.1x_1x_2+0.2x_2x_3 \leq 16$ $2x_1^2+x_2^2-2x_3^2 \geq 2$	$F=11.68$	$[-10 \quad -10 \quad -10]$	$[10 \quad 10 \quad 10]$
f_2	$\min F = x_1^2+x_2^2+2x_3^2+x_4^2-5x_1-5x_2-21x_3+7x_4$ s.t. $x_1^2+x_2^2+x_3^2+x_4^2+x_1-x_2+x_3-x_4-8 \leq 0$ $x_1^2+2x_2^2+x_3^2+2x_4^2-x_1-x_4-10 \leq 0$ $2x_1^2+x_2^2+x_3^2+2x_1-x_2-x_4-5 \leq 0$	$F=-44.00$	$[-10 \quad -10 \quad -10 \quad -10]$	$[10 \quad 10 \quad 10 \quad 10]$

Table 2
Results of PSO–LDW, PSO–NDW and IPSO algorithms for benchmark functions (population size of 20).

Function	Best results			Average results			Worst results		
	PSO–LDW	PSO–NDW	IPSO	PSO–LDW	PSO–NDW	IPSO	PSO–LDW	PSO–NDW	IPSO
f_1	11.6727	11.6752	11.6794	11.6497	11.6565	11.6755	11.5824	11.6062	11.6741
f_2	−43.9845	−43.9833	−43.9991	−43.9269	−43.9514	−43.9914	−43.9049	−43.9182	−43.9881

Table 3
Results of PSO–LDW, PSO–NDW and IPSO algorithms for benchmark functions (population size of 40).

Function	Best results			Average results			Worst results		
	PSO–LDW	PSO–NDW	IPSO	PSO–LDW	PSO–NDW	IPSO	PSO–LDW	PSO–NDW	IPSO
f_1	11.6745	11.6776	11.6796	11.6614	11.6687	11.6842	11.5946	11.6213	11.6773
f_2	−43.9893	−43.9911	−43.9992	−43.9311	−43.9732	−43.9936	−43.9267	−43.9239	−43.9915

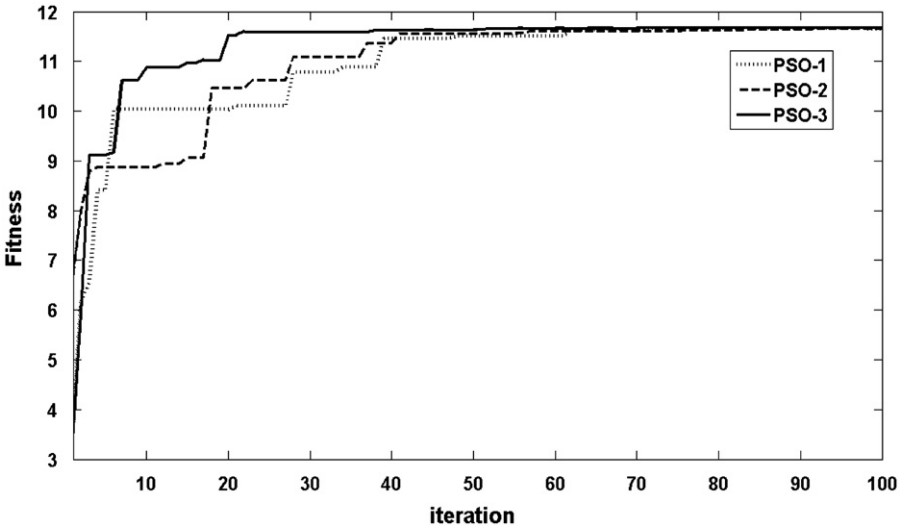

Fig. 1. Comparison of convergence of objective function for function 1.

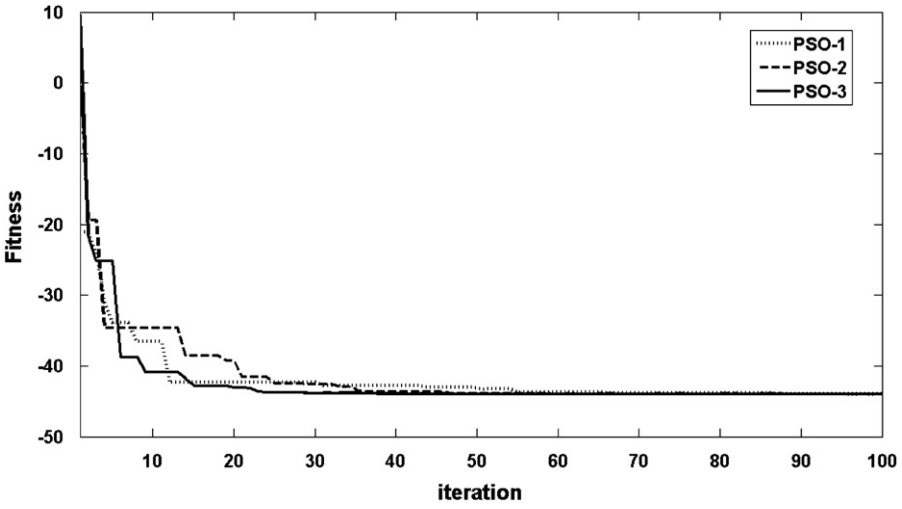

Fig. 2. Comparison of convergence of objective function for function 2.

by linearizing the constraints and approximating the Lagrangian function quadratically:

$$L(x,\lambda) = J(x) + \sum_{i=1}^{m} \lambda_i \psi_i(x) \tag{12}$$

At each iteration, an approximation of the Hessian of the Lagrangian function H_k is made.

The process starts from given iteration x_k, then, the following quadratic programming (QP) sub-problem is formed to solve:

$$\min \ \tfrac{1}{2} d^T H_k d + \nabla f(x_k)^T d \tag{13}$$

$$\nabla \psi_i(x_k)^T d + \psi_i(x_k) = 0, \ \ i = 1,\ldots,m_e \tag{14}$$

$$\nabla \psi_i(x_k)^T d + \psi_i(x_k) \geq 0, \ \ i = m_e,\ldots,m \quad d \in R^n \tag{15}$$

This sub-problem is a quadratic programming (QP) sub-problem whose solution is used to form a search direction for a line search procedure. In other words, the solution is used to form the next iterate:

$$x_{k+1} = x_k + \alpha_k d_k \tag{16}$$

The step length parameter α_k is determined by an appropriate line search procedure so that a sufficient decrease in a merit function is obtained. The method is vastly used in optimization problems, but it is also known that it depends on the initial estimate (Costa et al., 2005).

A number of authors have successfully applied SQP method to the solution of optimal control problems. In particular, Goh and Teo (1988) and Teo et al. (1991) approximate the control variable as piecewise constant and solve the NLP problem using the SQP technique.

6. Hybrid IPSO–SQP algorithm for optimal control

The proposed IPSO–SQP is an optimization algorithm combining an improved PSO (IPSO) algorithm with SQP algorithm, in order to solve NOC problems. The PSO algorithm is a global algorithm, which has a strong ability to find global optimistic result. However, it has a disadvantage that the search around global optimum is very slow. The SQP algorithm, on the contrary, has a strong ability to find local optimistic result for NOC problem, but its ability to find the global optimistic result is weak. Although

it has advantages in terms of computational robustness and their usefulness for practical problems, it is usually difficult to choose appropriate initial solutions.

By combining the IPSO with SQP, a new algorithm referred to as IPSO–SQP hybrid algorithm is formulated in this paper. Similar to the PSO algorithm, the IPSO–SQP algorithm's searching process is also started from initializing a group of random particles. First, IPSO algorithm is run to search the global best position in the solution space. Then SQP algorithm is used to search around the global optimum. In this way, this hybrid algorithm may find an optimum more quickly and accurately. The procedure for this IPSO–SQP algorithm can be summarized as follows:

Step 1: Initialize the positions and velocities of a group of particles randomly. Each particle is a potential solution for NOC problem in hand, i.e., a single particle represents the $m \times N$ elements of the stacked control vectors.
Step 2: Evaluate each initialized particle's fitness value, based on PI criteria mentioned by Eq. (4), including penalty functions.
Step 3: If the maximal iterative iterations are arrived, go to Step 7, else, go to Step 4.
Step 4: The best particle of the current particles is stored. If the change between the current best particle fitness value and its previous one is smaller than a predefined value, go to step 7, else continue.
Step 5: The positions and velocities of all the particles are updated according to Eqs. (6) and (7), and then a group of new particles is generated.
Step 6: Reduce the inertia weight for each particle according to Eq. (11) and go to step 2.
Step 7: Use SQP algorithm to search around global best, which is found by IPSO to find finer solutions. In this case, the best solution obtained by IPSO is considered as the initial guess for SQP algorithm.

7. Simulation results

In order to show the feasibility of the proposed IPSO–SQP for solving NOC problems, two benchmark problems are considered: (1) the continuous stirred-tank chemical reactor and (2) a mathematical system with nonlinear inequality constraint. Then, the performance of the proposed IPSO–SQP algorithm is compared with some heuristic algorithms used earlier for solving NOC problems, such as real-coded GA (Sarkar and Modak, 2004), DE (Lopez Cruz et al., 2003) and PSO (Herrera and Zhang, 2009). Three improved PSO algorithms are used, instead of standard PSO algorithm with a fixed inertia weight employed by Herrera and Zhang (2009): PSO–LDW, PSO–NDW and the proposed IPSO algorithm. The motivation is to show that the proposed IPSO algorithm is superior to the earlier well-known PSO algorithms for integrating with SQP algorithm. For all simulations, both c_1 and c_2 are set to 2 (Shi and Eberhart, 1998) for all PSO algorithms; in both PSO–TVIW and PSO–NTVIW algorithms $\omega_{\max}$ and $\omega_{\min}$ are set to 0.9 and 0.4, respectively, and the modulation index, n is set to 1.2 in PSO–NTVIW (Chatterjee and Siarry, 2006). For GA, the crossover and mutation rates are considered as 0.8 and 0.1, respectively (Grefenstette, 1986). For DE, all of its parameters are the same as in Lopez Cruz et al. (2003). All the methods are coded in Matlab 7.7 on PC with Pentium V, 7500 MHz/1024 MB RAM.

7.1. Continuous stirred-tank chemical reactor (CSTCR)

CSTCR is an NOC benchmark problem that has been used by several researchers (Luus and Cormack, 1972; Ali et al., 1997; Luus, 2000; Lopez Cruz et al., 2003; Bayón et al., 2009) to evaluate their methods. The state equations for a CSTCR are

$$\dot{x}_1 = -(2+u)(x_1+0.25)+(x_2+0.5)\exp\left(\frac{25x_1}{x_1+2}\right) \tag{17}$$

$$\dot{x}_1 = 0.5-x_2-(x_2+0.5)\exp\left(\frac{25x_1}{x_1+2}\right) \tag{18}$$

with initial condition $X(0)=[0.09 \quad 0.09]^T$. $x_1(t)$ is the deviation from the steady-state temperature, $x_2(t)$ is the deviation from the steady-state concentration and $u(t)$ is the normalized control variable that represents the effect of the flow-rate of the cooling fluid on chemical reactor. The objective is to determine the unconstrained $u_*(t)$ to minimize the quadratic performance measure:

$$J = \int_0^{0.78} [x(t)_1^2 + x(t)_2^2 + 0.1u^2(t)]dt \tag{19}$$

The performance measure indicates that the desired objective is to maintain the temperature and concentration close to their steady-state values without expending large amount of control effort.

This optimal control problem provides a good test problem for optimization procedures and is a member of the list of benchmark problems proposed in the handbook of test problems in local and global optimization (Floudas et al., 1999).

Ali et al. (1997) used eight stochastic global optimization algorithms to solve this problem and their results vary from $J=0.135$ to 0.245. Lopez Cruz et al. (2003) used four evolutionary algorithms (EA) and compared their results with the first order gradient method and the IDP. For the first-order gradient algorithm, they showed that its convergence to the local or global optimum highly depends on the initial values for the control. In fact, if the initial conditions are selected appropriately, then it converges to global optimum, precisely. Also they show that the CPU time used by IDP is quite long and it may still converge to the local optimum. Finally, they showed that the minimum obtained with four EA algorithms varies from $J=0.1358$ to 0.1449.

To solve this problem by means of the proposed method, the time interval $[t_0, t_f]$ is discretized in $N=13$ time intervals, as done by Lopez Cruz et al. (2003). The search process of GA, DE, PSO–LDW, PSO–NDW and IPSO algorithms is terminated when the change in fitness value is smaller than 0.0001 for 10 iterations. Also, the search process of PSO algorithm is switched to SQP method, when the change in fitness value is smaller than 0.0001 for 10 iterations

Tables 4 and 5 list the results obtained by each algorithm, where each algorithm is implemented 20 times independently for a population size of 20 and 40, respectively. The results indicate in how many iterations and the necessary time the convergence of the solution or success is met. The average of elapsed time in 20 runs is considered as a criterion for computational time.

Table 4
Comparison of PSO–LDW, PSO–NDW and IPSO in terms of accuracy and time required to find a solution for CSTCR problem (population size=20).

	Best results	Mean results	Worst results	Iteration	Elapsed time (s)
GA	0.14154	0.14979	0.15682	132.78	251.4885
DE	0.13943	0.14781	0.15138	81.45	48.3484
PSO–LDW	0.13892	0.14838	0.15034	99.12	44.7278
PSO–NDW	0.13984	0.14489	0.14953	78.94	34.8278
IPSO	0.13667	0.14083	0.14425	51.38	23.5064
IPSO–SQP	0.13549	0.13552	0.13554	75.14	25.2687

From these tables, the following results can be concluded. First of all, it is clearly obvious that IPSO algorithm has better solution accuracy and less computational time than GA, DE, PSO–LDW and PSO–NDW algorithms. So, the proposed IPSO algorithm is more appropriate than other algorithms for combining with SQP algorithm. Second, it is clear that the computational time for the proposed hybrid IPSO–SQP algorithm is considerably less than GA, DE, PSO-LDW and PSO–NDW algorithms. However, since in the hybrid IPSO–SQP algorithm, when the region of global optimum is reached by IPSO, the region is fine tuned by running some SQP iterations; the number of iterations for IPSO–SQP algorithm is more than IPSO algorithm. But, since the simulation time of SQP algorithm is small, the computational time of IPSO–SQP is very close to the computational time of IPSO algorithm. Third, it is apparent that IPSO–SQP is more accurate and more robust than other algorithms, since the worst and the best results obtained by the proposed IPSO–SQP algorithm are very close to each other. In fact, when SQP is integrated with the IPSO, it produces quality solutions as compared to the one produced by other algorithms.

Finally, comparing the results of the proposed method with other heuristic methods reported by Ali et al. (1997) and Lopez Cruz et al. (2003), it is clear that the proposed method is more robust and more accurate than other earlier reported methods. Fig. 3 shows the optimum control trajectory obtained by IPSO–SQP.

7.2. Mathematical system with nonlinear inequality constraint

This system involves a nonlinear inequality constraint and has been studied by several researchers (Mehra and Davis, 1972; Goh and Teo, 1988; Vlassenbroeck, 1988; Teo et al., 1991; Elnagar et al., 1995; Mekarapiruk and Luus, 1997). The state equations for the system are

$$\dot{x}_1 = x_2 \tag{20}$$

$$\dot{x}_2 = -x_2 + u \tag{21}$$

$$\dot{x}_3 = x_1^2 + x_2^2 + 0.005u^2 \tag{22}$$

with initial condition $X(0)=[0 \quad -1 \quad 0]^T$.

The nonlinear inequality constraint to be satisfied is

$$h(\boldsymbol{X}) = x_2 + 0.5 - 8(t-0.5)^2 \le 0 \tag{23}$$

The control is bounded by

$$-20 \le u \le 20 \tag{24}$$

The performance index to be minimized is

$$J = x_3(t_f) \tag{25}$$

where $t_f=1$. To solve this problem by means of the proposed method, the time interval $[t_0, t_f]$ is discretized in $N=20$ time intervals as done by Mekarapiruk and Luus (1997).

Goh and Teo (1988) solved this problem using the control parameterization technique, and the result obtained was $J=0.1816$. Mekarapiruk and Luus (1997) proposed a penalty function and solved this inequality state constraint. They obtained a result of $J=0.1769$.

Again, to show the superiority of the proposed algorithm, the performance of the IPSO–SQP algorithm is compared with GA, DE, PSO–LDW, PSO–NDW and IPSO algorithms. Tables 6 and 7 list the

Table 5
Comparison of PSO–LDW, PSO–NDW and IPSO in terms of accuracy and time required to find a solution for CSTCR problem (population size=40).

	Best results	Mean results	Worst results	Iteration	Elapsed time (s)
GA	0.14096	0.14693	0.15316	118.45	409.4908
DE	0.13804	0.14548	0.14913	77.82	65.6510
PSO–LDW	0.13814	0.14611	0.14947	82.35	64.0209
PSO–NDW	0.13765	0.14376	0.14843	68.62	45.5656
IPSO	0.13652	0.13955	0.14509	33.47	28.8921
IPSO–SQP	0.13549	0.13551	0.13555	52.68	31.3467

Table 6
Comparison of PSO–LDW, PSO–NDW and IPSO in terms of accuracy and time required to find a solution for system with nonlinear inequality constraint (population size=40).

	Best results	Mean results	Worst results	Iteration	Elapsed time (s)
GA	0.20885	0.27861	0.31069	311.28	1107.9487
DE	0.19854	0.23877	0.29838	184.95	147.6557
PSO–LDW	0.21929	0.25431	0.30194	254.93	170.8563
PSO–NDW	0.19761	0.22635	0.26474	195.84	151.8745
IPSO	0.18153	0.19167	0.20643	141.54	104.7271
IPSO–SQP	0.17277	0.17283	0.17286	169.95	107.3312

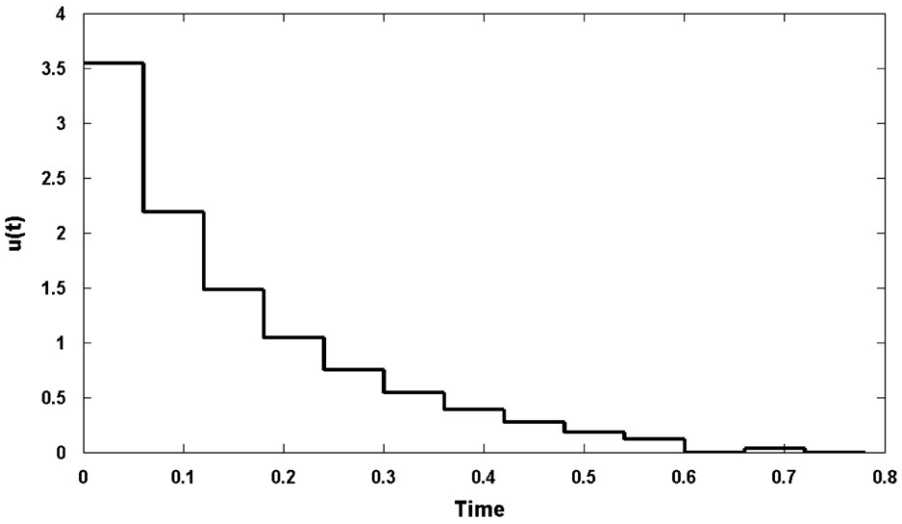

Fig. 3. Optimum control trajectory for CSTCR problem.

results obtained by each algorithm, where each algorithm is implemented 20 times independently for a population size of 40 and 60, respectively.

From Tables 6 and 7, again it is clear that the IPSO has less computational time and also better solution accuracy as compared with GA, DE, PSO–LDW and PSO–NDW algorithms and hence it is more proper for integrating with SQP algorithm. Also, the computational time of the proposed hybrid IPSO–SQP algorithm is less than GA, DE, PSO–LDW and PSO–NDW algorithms and is a bit more than IPSO algorithm. Once again, it is obvious that the hybrid IPSO–SQP algorithm is more robust and accurate than GA, DE, PSO–LDW, PSO–NDW and IPSO algorithms. Finally, comparing the results of the proposed method with Goh and Teo (1988) and Mekarapiruk and Luus (1997), it is concluded that the proposed method has better accuracy than others.

Fig. 4 shows the optimum control trajectory obtained by IPSO–SQP. The trajectory of the function of state $h(\boldsymbol{X})$ is shown in Fig. 5. As can be seen, the constraint is satisfied throughout the time interval.

Table 7
Comparison of PSO–LDW, PSO–NDW and IPSO in terms of accuracy and time required to find a solution for system with nonlinear inequality constraint (population size = 60).

	Best results	Mean results	Worst results	Iteration	Elapsed time (s)
GA	0.20172	0.25496	0.30525	294.95	1721.9487
DE	0.19523	0.22106	0.26487	169.95	192.1882
PSO–LDW	0.20854	0.23758	0.29561	232.23	220.2269
PSO–NDW	0.19267	0.21759	0.24893	174.92	167.2037
IPSO	0.17781	0.18945	0.20274	123.87	124.0275
IPSO–SQP	0.17276	0.17281	0.17284	145.29	126.2628

8. Conclusion

To solve NOC problems, we proposed a method based on combination of an improved PSO (IPSO) algorithm and successive quadratic programming (SQP) algorithm, namely IPSO–SQP. We showed that the hybrid method has the advantage of both IPSO and SQP methods while does not inherent their drawbacks. As the IPSO algorithm successfully searches all space during the initial stages of a global search, we used IPSO algorithm at earlier stage of IPSO–SQP. As long as the change in the fitness of global optimum is less than a predefined value, the algorithm switches to SQP to find an accurate solution. The results of the proposed hybrid method were compared with some heuristic optimization

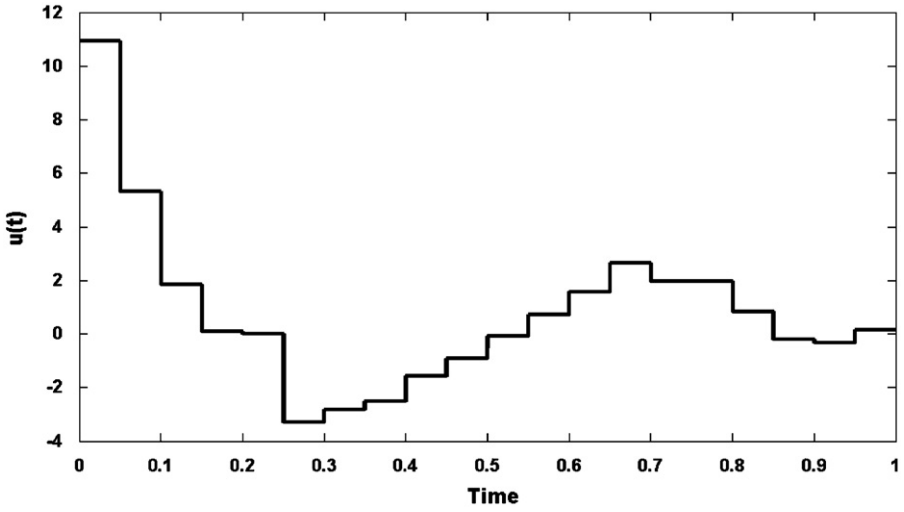

Fig. 4. Optimum control trajectory for system with nonlinear inequality constraint problem.

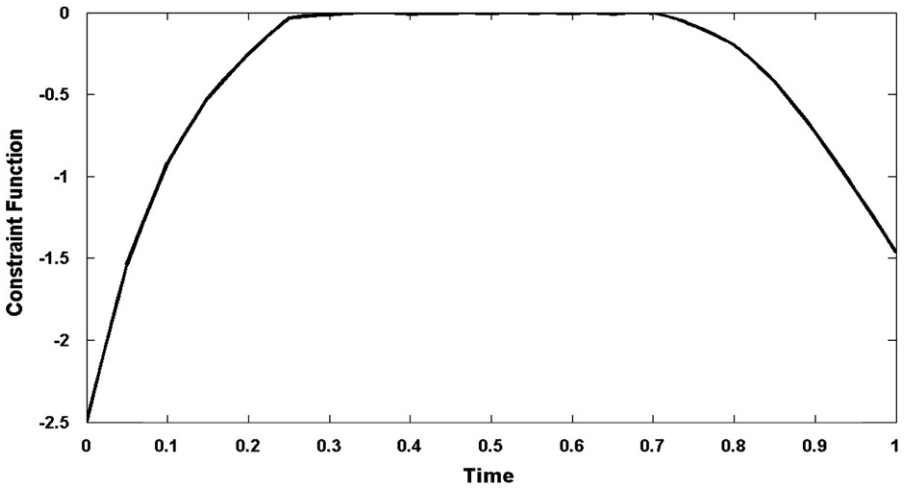

Fig. 5. Trajectory of the function $h(x)$ in system with nonlinear inequality constraint after applying optimal control policy.

algorithms such as GA, DE, PSO–LDW, PSO–NDW and IPSO, on two NOC problems. The results showed that the proposed hybrid method is more robust and accurate than other heuristic algorithms.

References

Agrawal, S.K., Fabien, B.C., 1999. Optimization of Dynamic Systems, Solid Mechanics and its Applications. Kluwer Academic Publishers, Dordrecht, The Netherlands.

Ali, M.M., Storey, C., Törn, A., 1997. Application of stochastic global optimization algorithms to practical problems. Journal of Optimization Theory and Applications 95, 545–563.

Arumugam, M.S., Rao, M.V.C., 2008. On the improved performances of the particle swarm optimization algorithms with adaptive parameters, cross-over operators and root mean square (RMS) variants for computing optimal control of a class of hybrid systems. Applied Soft Computing 8, 324–336.

Babu, B.V., Angira, R., 2006. Modified differential evolution (MDE) for optimization of non-linear chemical processes. Computers and Chemical Engineering 30, 989–1002.

Bayón, L., Grau, J.M., Ruiz, M.M., Suárez, P.M., 2009. Initial guess of the solution of dynamic optimization of chemical processes. Journal of Mathematical Chemistry. doi:10.1007/s10910-009-9614-5.

Bryson, J.A., Ho, Y.C., 1975. Applied Optimal Control: Optimization, Estimation and Control. Hemisphere, Washington, DC.

Bryson, A.E., 1999. Dynamic Optimization. Addison-Wesley, Menlo Park, NY.

Chatterjee, A., Siarry, P., 2006. Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization. Computers and Operations Research 33 (3), 859–871.

Chambers, L., 1995. Practical Handbook of Genetic Algorithms: New Frontiers, vol. II. CRC Press.

Chang, W., 2007. Nonlinear system identification and control using a real-coded genetic algorithm. Applied Mathematical Modeling 31, 541–550.

Clerc, M., Kennedy, J., 2002. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation 6 (1), 58–73.

Coello, C.A.C., 2000. Use of a self-adaptive penalty approach for engineering optimization problems. Computers in Industry 41, 113–127.

Costa, C.B.B., da Costa, A.C., Maciel Filho, R., 2005. Mathematical modeling and optimal control strategy development for an adipic acid crystallization process. Chemical Engineering and Processing 44, 737–753.

Elnagar, G., Kazemi, M.A., Razzaghi, M., 1995. The Pseudospectral Legendre Method for discretizing optimal control problems. IEEE Transactions on Automatic Control 40, 1793–1796.

Floudas, C.A., Pardalos, P.M., Adjiman, C.S., Esposito, W.R., Gumiis, Z.H., Harding, S.T., Klepeis, J.L., Meyer, C.A., Schweiger, C.A., 1999. Handbook of test problems in local and global optimization, Series in Nonconvex Optimization and its Applications. Kluwer Academic Publishers, Dordrech.

Goh, C.J., Teo, L.K., 1988. Control parameterization: a unified approach to optimal control problems with general constraints. Automatica 24, 3–18.

Goldberg, D., 1989. Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading, MA.

Grefenstette, J.J., 1986. Optimization of control parameters for genetic algorithms. IEEE Transactions on Systems, Man and Cybernetics 16 (1), 122–128.

Herrera, F., Zhang, J., 2009. Optimal control of batch processes using particle swam optimization with stacked neural network models. Computers and Chemical Engineering 33, 1593–1601.

Holland, J.H., 1975. Adaptation in Natural and Artificial Systems. The University of Michigan Press, Ann Arbor, Michigan.

Homaifar, A., Qi, C.X., Lai, S.H., 1994. Constrained optimization via genetic algorithms. Simulation 62 (4), 242–254.

Joines, J.A., Houck, C.R., 1994. On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GAs, Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE Press, Orlando, pp. 579–584.

Kennedy, J., Eberhart, R.C., 1995. Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks, vol. 4. pp. 1942–1948.

Kennedy, J., Eberhart, R.C., Shi, Y., 2001. Swarm Intelligence. Morgan Kaufmann Publishers, San Francisco.

Lopez Cruz, I.L., Van Willigenburg, L.G., Van Straten, G., 2003. Efficient differential evolution algorithms for multimodal optimal control problems. Applied Soft Computing 3, 97–122.

Luus, R., Cormack, D.E., 1972. Multiplicity of solutions resulting from the use of variational methods in optimal control problems. Canadian Journal of Chemical Engineering 50, 309–312.

Luus, R., 2000. Iterative Dynamic Programming. Chapman and Hall/CRC Press, Boca Raton, FL.

Mathur, M., Sachin, K., Sidhartha, P., Jayaraman, V., Kulkarni, B.D., 2000. Ant colony approach to continuous functions optimization. Industrial & Engineering Chemistry Research 39, 3814–3822.

Mehra, R.K., Davis, R.E., 1972. A generalized gradient method for optimal control problems with inequality constraints and singular arcs. IEEE Transactions on Automatic Control 17, 69–78.

Mekarapiruk, W., Luus, R., 1997. Optimal control of inequality state constrained systems. Industrial & Engineering Chemistry Research 36, 1686–1694.

Modares, H., Alfi, A., Fateh, M.M., 2010a. Parameter identification of chaotic dynamic systems through an improved particle swarm optimization. Expert Systems with Applications 37, 3714–3720.

Modares, H., Alfi, A., Naghibi Sistani, M.B., 2010b. Parameter estimation of bilinear systems based on an adaptive particle swarm optimization. Engineering Applications of Artificial Intelligence. doi:10.1016/j.engappai.2010.05.003.

Onwubolu, G.C., Babu, B.V., 2004. New Optimization Techniques in Engineering. Springer-Verlag, Heidelberg, Germany.

Price, K., Storn, R., 1997. Differential evolution, a simple evolution strategy for fast optimization. Dr. Dobb's Journal 1997 (22), 18–24.

Sarkar, D., Modak, J.M., 2004. Optimization of fed-batch bioreactors using genetic algorithm: multiple control variables. Computers and Chemical Engineering 28, 789–798.

Sewald, H., Kumar, R.R., 1995. Genetic algorithm approach for optimal control problems with linearity appearing controls. Journal of Guidance, Control and Dynamics 18 (1), 177–182.

Shi, Y., Eberhart, R.C., 1998. A modified particle swarm optimizer. In: Proceedings of the Conference on Evolutionary Computation. pp. 69–73.

Stryk, O.V., Bulirsch, R., 1992. Direct and indirect methods for trajectory optimization. Annals of Operations Research 37, 357–373.

Teo, K.L., Goh, C.J., Wong, K.H., 1991. A Unified Computational Approach to Optimal Control Problems. Wiley, New York, pp. 146–147.

Varadarajan, M., Swarup, K.S., 2008. Differential evolution approach for optimal reactive power dispatch. Applied Soft Computing 8, 1549–1561.

Victoire, T.A., Jeyakumar, A.E., 2004. Hybrid PSO–SQP for economic dispatch with valve-point effect. Electric Power Systems Research 71, 51–59.

Vlassenbroeck, J., 1988. A Chebyshev polynomial method for optimal control with state constraints. Automatica 24 (4), 499–506.

Yamashita, Y., Shima, M., 1997. Numerical computational method using genetic algorithms for the optimal control problem with terminal constraints and free parameters. Non-linear Analysis: Theory, Methods and Applications 30 (4), 2285–2290.