Stability of Nonlinear Systems Using Optimal Fuzzy Controllers and Its Simulation by Java Programming

Mohammad Jawad Mahmoodabadi and S. Arabani Mostaghim

***Abstract*—In this paper, at first, the single input rule modules (SIRMs) dynamically connected fuzzy inference model is used to stabilize a double inverted pendulum system. Then, a multi-objective particle swarm optimization (MOPSO) is implemented to optimize the fuzzy controller parameters in order to decrease the distance error of the cart and summation of the angle errors of the pendulums, simultaneously. The feasibility and efficiency of the proposed Pareto front is assessed in comparison with results reported in literature and obtained from other algorithms. Finally, the Java programming with applets is utilized to simulate the stability of the nonlinear system and explain the internet-based control.**

***Index Terms*—Multi-objective algorithm, particle swarm optimization, fuzzy control, Java programming, double inverted pendulum system.**

I. Introduction

RECENTLY, the internet technologies such as Java Applets have been widely used for designing simulation programs for complicated systems. In [1], the simulation of fuzzy control for an inverted pendulum system based on Java Applets is presented. Mahmoodabadi et al. applied Java programming to internet-based explanation of state feedback control for dynamic systems, e.g., the inverted pendulum and ball-beam system [2].

On the other hand, the development of high-performance controllers for various complex problems has been a major research topic among the control engineering practitioners in recent years. For example, Liu *et al*. proposed a type-2 fuzzy switching control system for biped robots [3]. Tseng and Chen designed a robust fuzzy controller for nonlinear discrete-time systems with persistent bounded disturbances [4]. Huang *et al*. designed and implemented a fuzzy controller for a two-wheel inverted pendulum system [5]. Zhu applied a fuzzy optimal controller for multistage fuzzy systems [6]. Wang *et al*. proposed an adaptive fuzzy backstepping controller for a class of nonlinear systems with sampled and delayed measurements [7].

In this way, synthesis of control policies has been presented as optimization problems of certain performance measures of the controlled systems. A very effective means of solving such optimum controller design problems is evolutionary algorithms. Yi *et al*. applied single input rule modules dynamically connected fuzzy inference models for stabilizing a double inverted pendulum system and tuned parameters by a random optimization method [8]. Shook *et al*. used genetic algorithm to optimize fuzzy logic controllers which were designed to manage two dampers for mitigation of seismic loads [9]. The membership functions of multi-stage fuzzy load frequency control were automatically designed by particle swarm optimization in [10]. A 2DOF planer robot has been controlled by fuzzy logic controller and particle swarm optimization has been utilized to tune fuzzy parameters in [11].

However, the design criteria (objective functions) of these practical engineering problems may conflict with each other so that improving one of them will deteriorate another. This inherent conflicting behavior of such objective functions leads to a set of optimal solutions named Pareto. This type of problems can be solved by using the multi-objective evolutionary algorithms. For instance, Ahmadi *et al*. applied a new multi-objective uniform-diversity genetic algorithm with a diversity preserving mechanism to optimization a robust reliability-based controller with probabilistic uncertainties [12]. Mahmoodabadi *et al*. used multi-objective particle swarm optimization for optimum design of fuzzy controllers for nonlinear systems [13].

One of the modern heuristic algorithms is Particle Swarm Optimization (PSO) introduced by Kennedy and Eberhart [14], which is developed through simulation of simplified social systems and is robust for solving nonlinear optimization problems [15]. The PSO technique can generate a high quality solution with less calculation time and a more stable convergence characteristic in comparison with other evolutionary algorithms [16]. These advantages make PSO widely used in many engineering applications. In the recent years, several state-of-the-art PSO variants have been proposed, such as those in [17]−[23]. Furthermore, recently, several approaches, such as dynamic neighborhood PSO [24], dominated tree[25], Sigma method [26], vector evaluated PSO [27], dynamic multiple swarms [28], dynamic population size and adaptive local archives [29], etc., have been proposed to extend the PSO algorithm to deal with multi-objective optimization problems.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

This article was recommended by Associate Editor Huijun Gao.

Citation: M. J. Mahmoodabadi and S. A. Mostaghim, "Stability of nonlinear systems using optimal fuzzy controllers and its simulation by Java programming," *IEEE/CAA Journal of Automatica Sinica*, pp. 1−10, 2017. DOI: 10.1109/JAS.2017.7510388.

M. J. Mahmoodabadi is with the Department of Mechanical Engineering, Sirjan University of Technology, Sirjan, Iran(e-mail: Mahmoodabadi@sirjantech.ac.n).

S. A. Mostaghim is with the Department of Computer Engineering, Faculty of Engineering, University of Guilan, Rasht, Iran(e-mail: saideh_arabani@yahoo.com).

This paper is aimed at improving multi-objective particle swarm optimization (MOPSO) by increasing the rate of convergence and diversity of solutions. The proposed MOPSO is applied to Pareto optimal fuzzy control design of a doubled inverted pendulum system. In the other words, by using the MOPSO algorithm, the important parameters of the single input rule modules dynamically connected fuzzy inference model are optimized in order to simultaneously decrease the distance error of the cart and summation errors of angles for the shorter and longer pendulums.

II. Basic Concepts of Fuzzy Controller

Yubazaki *et al*. originally proposed single input rule modules (SIRMs) dynamically connected fuzzy inference model for plural input fuzzy control [30]. In this controller, for each input item (the states of the system), a SIRM and a dynamic importance degree (DID) are defined. DID consists of a base value that insures the role of the input item through a control process, and a dynamic value that changes with control situations to adjust DID. In the following, the concepts of the SIRM and DID are described.

III. SIRM

Consider a dynamic system with n input items and 1 output item. Since there are n input items, n SIRMs would be defined as:

$$\text{SIRM}_i: \quad \{R_i^j: \text{ if } x_i = A_i^j \text{ then } u_i = C_i^j\}_{j=1}^{m_i}\}, \quad (1)$$

where SIRM_i means the SIRM referred to the i-th input item, and R_i^j is the j-th rule in the SIRM_i. x_i corresponds to the variable in the antecedent part, and u_i is the variable in the consequent part. A_i^j and C_i^j are the membership functions of the x_i and u_i in the j-th rule of the SIRM_i, respectively. $j = 1, 2, \cdots, m$ is the index number of the rules in the SIRM_i. Here, the rules and membership functions for SIRMs to stabilize the double inverted pendulum system are according to Reference [8].

IV. DID

To express the different role of each input item in system performance, the dynamic importance degree w_i^D for each input item x_i $(i = 1, 2, \cdots, n)$is defined as follows:

$$w_i^D = w_i^0 + B_i \Delta w_i^0, \quad (2)$$

where w_i^0 is a base value, Δw_i^0 is a fuzzy variable, and B_i is a breadth for the ith input item $(i = 1, 2, \cdots, n)$. In this paper, the fuzzy variable Δw_i^0 of the inputs x_i $(i = 1, 2, \cdots, n)$ to stabilize the double inverted pendulum system is completely similar to [8].

Once SIRMs and DID are determined, the driving force F could be obtained by following equation:

$$F = OSF \times u = OSF \times \sum_{i=1}^{n} w_i^D u_i^0, \quad (3)$$

where OSF is the output scaling factor, u_i^0 is the fuzzy inference result obtained from SIRM_i.

In this paper, the base values w_i^0 $(i = 1, 2, \cdots, n)$, the breadths B_i $(i = 1, 2, \cdots, n)$, and the output scaling factor OSF would be determined by using the multi-objective particle swarm optimization algorithm.

V. Double Inverted Pendulum

In Fig. 1, the structure of a double inverted pendulum shows that the state vector $x = [x_1, x_2, x_3, x_4, x_5, x_6]^{\mathrm{T}} = [\alpha, \dot{\alpha}, \beta, \dot{\beta}, x, \dot{x}]^{\mathrm{T}}$ includes the angle and angular velocity of the longer pendulum, the angle and angular velocity of the shorter pendulum, the position and velocity of the cart. The dynamical equations of the system are as follows:

$$\begin{aligned}
&(M + m_1 + m_2)\ddot{x} + m_1 l_1 \ddot{\alpha}\cos(\alpha) + m_2 l_2 \ddot{\beta}\cos(\beta) \\
&\quad = F + m_1 l_1 \dot{\alpha}^2 \sin(\alpha) + m_2 l_2 \dot{\beta}^2 \sin(\beta), \\
&m_1 l_1 \ddot{x}\cos(\alpha) + \frac{4 m_1 l_1^2}{3}\ddot{\alpha} = m_1 l_1 g \sin(\alpha), \\
&m_2 l_2 \ddot{x}\cos(\beta) + \frac{4 m_2 l_2^2}{3}\ddot{\beta} = m_2 l_2 g \sin(\beta). \quad (4)
\end{aligned}$$

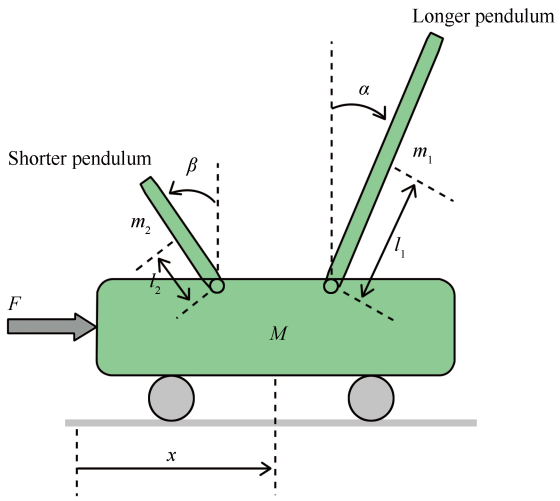

Fig. 1. The structure of a double inverted pendulum system.

The masses of the cart, longer pendulum, and shorter pendulum are M, m_1 and m_2, respectively. g represents the gravity acceleration. l_1 and l_2 are half length of the longer and shorter pendulums, respectively. The driving force applied horizontally to the cart is denoted as F. For simulation, the following specifications are used. $M = 1$, $m_1 = 0.3\,\text{kg}$, $m_2 = 0.1\,\text{kg}$, $l_1 = 0.6\,\text{m}$, $l_2 = 0.2\,\text{m}$, and $g = 9.8\,\text{m/s}^2$. Furthermore, The initial and desired values are $x = [x_1, x_2, x_3, x_4, x_5, x_6] = [\pi/36, 0, \pi/36, 0, 0, 0]$ and $x = [x_1, x_2, x_3, x_4, x_5, x_6] = [0, 0, 0, 0, 0, 0]$, respectively.

VI. Basic Concepts of Optimization

A. Particle Swarm Optimization

Particle swam optimization is a population-based evolutionary algorithm that the candidate solutions are called particles and the position of each particle is changed via its own experience and its neighbors according to the following equations [31]−[34]:

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \overrightarrow{v}_i(t+1), \tag{5}$$

$$\overrightarrow{v}_i(t+1) = W\overrightarrow{v}_i(t) + C_1 r_1(\vec{x}_{pbest_i} - \vec{x}_i(t)) + C_2 r_2(\vec{x}_{gbest} - \vec{x}_i(t)), \tag{6}$$

where $\vec{x}_i(t)$ and $\vec{v}_i(t)$ denote the position and velocity of particle i, at time step t. $r_1, r_2 \in [0,1]$ are random values. C_1 are C_2 are the cognitive and social learning factors, respectively, and W is the inertia weight. The personal best position of the particle i is $\vec{x}_{pbest_i}$ and $\vec{x}_{gbest}$ is the position of the best particle of the entire swarm.

Experimental results illustrated that the linearly decreasing inertia weight over the iterations improves the performance of PSO [35]. Furthermore, it has shown that best solutions were determined when C_1 is linearly decreased and C_2 is linearly increased over the iterations [36]. Hence, the following linear formula for the inertia weight and learning factors are used.

$$W = W_1 - (W_1 - W_2) \times \left(\frac{t}{\text{maximum iteration}}\right), \tag{7}$$

$$C_1 = C_{1i} - (C_{1i} - C_{1f}) \times \left(\frac{t}{\text{maximum iteration}}\right), \tag{8}$$

$$C_2 = C_{2i} - (C_{2i} - C_{2f}) \times \left(\frac{t}{\text{maximum iteration}}\right), \tag{9}$$

where W_1 and W_2 are the initial and final values of the inertia weight, respectively. C_{1i}, C_{2i}, C_{1f}, and C_{2f} are the initial and final values of the learning factors, respectively. t is the current iteration number and maximum iteration is the maximum number of allowable iterations.

B. Multi-objective Particle Swarm Optimization

In the following, the selection strategies of global best position and personal best position for multi-objective PSO are described. Furthermore, a turbulence operator and a dynamic elimination method are introduced in order to enhance the solution searching ability of particles and to maintain the diversity of non-dominated solutions.

1) Selection of Global Best Position: A leader should be selected as $\vec{x}_{gbest}$ for each particle in order to update its position. Here, a leader selection technique based on density measures is used. In fact, a neighborhood radius ($R_{\text{neighborhood}}$) is considered for all non-dominated solutions. Two non-dominated solutions are neighbors if Euclidean distance (measured in the objective domain) between them is less than $R_{\text{neighborhood}}$. Using this definition, the neighbors of all non-dominated solutions are calculated in the objective function domain, and the particle with fewer neighbors is considered as the leader.

2) Selection of Personal Best Position: The Sigma method proposed in [26] is applied to determining the personal best position of a particle. After assigning parameter σ to all particles in the population and archive (10), particle k in the archive would be selected as the personal best position for particle i in the population, if the distance between σ_i and σ_k is minimum.

$$\sigma = \frac{f_1^2 - f_2^2}{f_1^2 + f_2^2}, \tag{10}$$

where f_1 and f_2 are two objective functions of the considered particle.

3) Turbulence Operator: In order to escape from local minima and find better solutions, a turbulence operator is used. N particles from the population are randomly chosen to perform that according to the following equation:

$$\vec{x}_i(t) = \vec{x}_{\min} + \text{rand} \times (\vec{x}_{\max} - \vec{x}_{\min}), \tag{11}$$

where $rand$ is a random number which generated uniformly in the interval [0, 1]. $\vec{x}_{\max}$ and $\vec{x}_{\min}$ are upper and lower bounds of the search space respectively. In this paper, $N = P_m \times$ number of particles where P_m is the probability of the turbulence operator.

4) Dynamic Elimination Technique: Elimination techniques are used to prune the archive and limit its size [37-40]; and here, the method introduced in [41] is applied. In this approach, for each particle in the archive, an elimination radius equal to $\varepsilon_{\text{Dynamic}}$ is defined. If Euclidean distance (in the objective function space) between two particles is less than $\varepsilon_{\text{Dynamic}}$, then one of them will be eliminated [41].

5) Details of the Proposed algorithm: Initially, the particles of the population should be randomly generated. Then, the inertia weight, the learning factors, and turbulence probability can be determined. In each iteration, the archive can be formed, and $\overrightarrow{x}_{pbest_i}$ and $\overrightarrow{x}_{gbest}$ are assigned. Then, for each particle, a random number $\rho \in [0,1]$ will be considered. If a particle has $\rho <$ (turbulence probability), then a new particle will be produced by the turbulent operator. Other particles that are not selected for turbulent operation will be enhanced by PSO. This cycle should be repeated until the criterion of the maximum iteration is reached. The pseudo code of this algorithm is illustrated in Table I.

VII. Pareto Design of SIRMs Dynamically Connected Fuzzy Controller

In this section, Pareto design of SIRMs dynamically connected fuzzy controllers using multi-objective particle swarm optimization is performed for the double inverted pendulum system. The distance error of the cart and the summation errors of angles for the shorter and longer pendulums are regarded as the objective functions that should be minimized simultaneously. The vector [w_1^0, w_2^0, w_3^0, w_4^0, w_5^0, w_6^0, B_1, B_2, B_3, B_4, B_5, B_6, OSF] is the vector of design variables where w_i^0 $(i = 1,2,3,4,5,6)$, B_i $(i = 1,2,3,4,5,6)$, and OSF are the base values, the breadths, and the output scaling factor, respectively. To compare the performance of the proposed method, two well-known versions of multi-objective optimization algorithms are used (Table II). Also, the configuration of the MOPSO algorithm is as follows: population size = 100, maximum iteration = 1000, $W_1 = 0.9$, $W_1 = 0.4$, $C_{1i} = 2.5$, $C_{1f} = 0.5$, $C_{2i} = 0.5$, $C_{2f} = 2.5$, $R_{\text{neighborhood}} = 0.02$, $P_m = 5/t$.

When the multi-objective optimization algorithms are applied to this problem, Pareto fronts will be achieved that are demonstrated in Fig. 2. In Fig. 2, Points A and C stand for the minimum distance error of the cart and the summation errors of angles for the shorter and longer pendulums, respectively.

TABLE I

THE PSEUDO CODE OF THE USED MULTI-OBJECTIVE PARTICLE SWARM OPTIMIZATION

Algorithm 1 Initialize population and the algorithm configuration of the proposed MOPSO.

For $t = 1$: maximum iteration
  Assign the inertia weight, the learning factors, and turbulence probability.
  Calculate the fitness values of all particles, update and prune the archive with respect to $\varepsilon_{Dynamic}$.
  Determine $\boldsymbol{x}_{gbest}(t)$.
  For $i = 1$ population size
    Determine $\boldsymbol{x}_{pbest_i}(t)$.
    $\rho = rand$
    If $\rho < (turbulenceprobability)$ then update position $\boldsymbol{x}_i(t)$ using turbulent operators.
      Else update velocity and position $\boldsymbol{x}_i(t)$ using PSO formulations.
    End.
  End.
End.

TABLE II

THE PARAMETER CONFIGURATIONS OF THE USED MULTI-OBJECTIVE ALGORITHMS IN THE COMPARISON

Algorithm	Population size	Technique of the archive	Crossover function (rate)	Mutation for pruning	Function function (rate)	Reference evaluations
Sigma MOPSO	100	Clustering	–	Uniform (0.1)	100 000	[26]
NSGAII	100	Crowding-distance	Scattered (0.8)	Uniform (0.1)	100 000	[42]

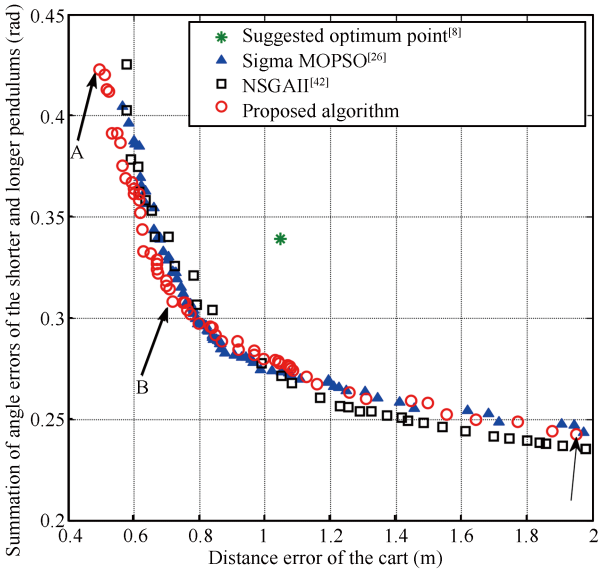

Fig. 2. The suggested optimum point by [8] and the obtained Pareto fronts by using of Sigma method [26], NSGAII [42], and proposed method for optimal design of SIRMs dynamically connected fuzzy controllers.

It is clear from Fig. 2 that some points obtained by the proposed algorithm dominate to the suggested optimum point by [8] (such as Point B). Furthermore, the point A has the best distance control of the cart and the worst angular control of the pendulums, while Point C has the best angular control of the pendulums and the worst distance control of the cart. Design variables, objective functions, and maximum driving forces corresponding to the optimum design points are presented in Table III. The time responses of cart and pendulums related to these points are shown in Figs. 3 – 5. Furthermore, Fig. 6 shows the driving forces of the optimum design points.

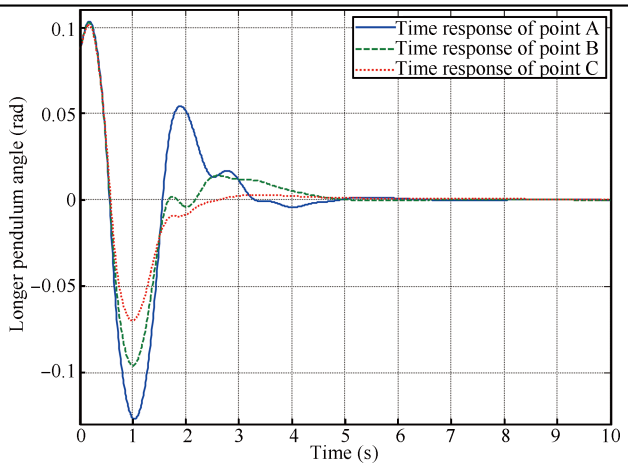

Fig. 3. Time responses of the longer pendulum for optimum design points A, B, and C shown in the Pareto front.

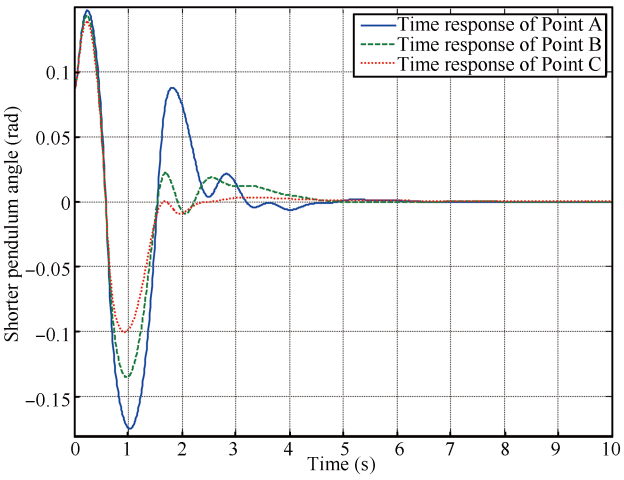

Fig. 4. Time responses of the shorter pendulum for optimum design Points A, B, and C shown in the Pareto front.

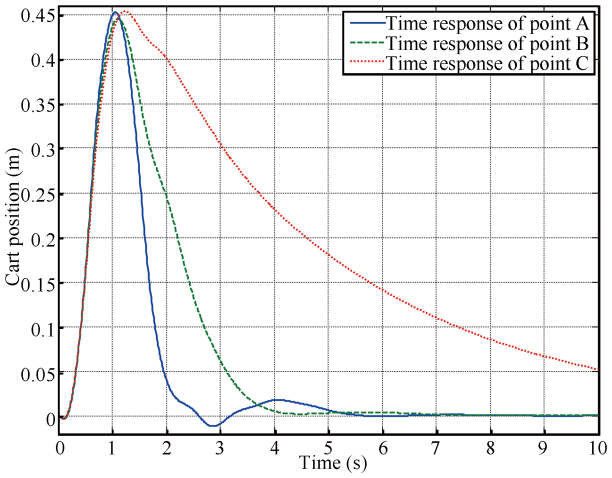

Fig. 5. Time responses of the cart for optimum design Points A, B, and C shown in the Pareto front.

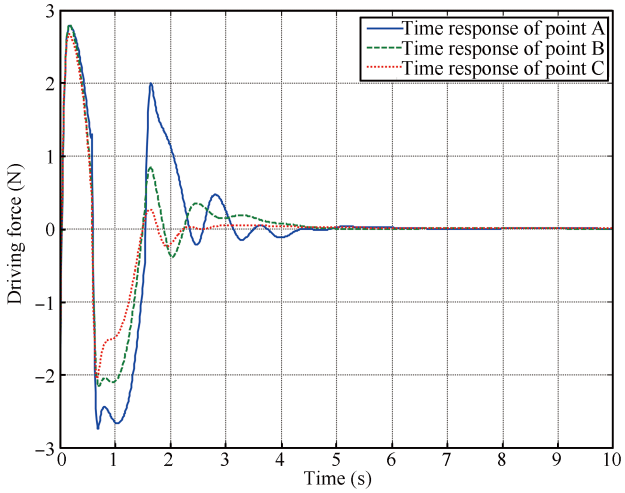

Fig. 6. Driving forces for optimum design Points A, B, and C shown in the Pareto front.

TABLE III
THE VALUES OF OBJECTIVE FUNCTIONS AND THEIR ASSOCIATED DESIGN VARIABLES FOR THE OPTIMUM POINTS SHOWN IN FIG. 2

Optimum design points	A	B	C
w_1^0	2.1641	2.1836	2.3155
w_2^0	4.6814	4.5599	4.5901
w_3^0	1.6095	1.6606	1.8544
w_4^0	1.7101	1.7806	1.8062
w_5^0	0.6906	0.4942	0.1584
w_6^0	0.3902	0.2728	0.2391
B_1	0.3754	0.4749	0.5054
B_2	0.3731	0.2781	0.1263
B_3	0.2986	0.3211	0.2213
B_4	1.4978	1.3056	1.3865
B_5	0.0000	0.0000	0.0000
B_6	0.1055	0.1157	0.0851
OSF	14.4369	14.5621	14.7365
The distance error of the cart (m)	0.4927	0.7183	1.9520
The summation of angle errors of the pendulums (rad)	0.4228	0.3082	0.2428
Maximum driving force (N)	2.7922	2.7954	2.7302

VIII. JAVA PROGRAMMING FOR STABILITY SIMULATION OF DOUBLE INVERTED PENDULUM SYSTEM

Java is an objected-oriented programming language, a run-time system, a set of development tools, and an application programming interface (API). The relationships among these elements are described in [1], [2].

Here, the Java applet is used to create the double inverted pendulum system and its SIRM fuzzy controller based on the proposed multi-objective particle swarm optimization. In fact, the coding of stability simulation via a computational tool as Java is to verify the obtained results in the previous section and education of internet-based control [1].

In the following, the main concepts of the Java programming are reviewed [43]−[45].

1) Classes, Methods, and Packages Java programs consisted of classes included methods. Java has many predefined named packages.

2) Java Development Environment Java programs mainly have five phases: edit, compile, load, verify, and execute.

3) Inheritance In Java, inheritance is a compile-time mechanism that extends a class to another class.

4) Applet An applet program written in the Java programming language can be placed in an HTML page and has five methods: init, start, paint, stop, and destroy.

5) Multithreading. Multithreaded programs support more than one concurrent thread of execution and able to execute the multiple sequences of instructions simultaneously.

For coding of stability simulation via Java, a package named JavaDoubleInvPend is defined. This package includes three classes named MainDIPClass, DIDClass, and SIRMClass as shown in Tables IV-VI, respectively. Two classes, DIDClass and SIRMClass are included methods that implement the equations in Sections II-A and II-B. MainDIPClass is one of Applet's subclasses and can inherit the Applet properties. Moreover, a method named DoubleInvPendSys() is defined in order to use the dynamic equations introduced in Section III to achieve the position of the cart and pendulums (Table VII). Hence, by using of these classes, we could achieve the position of the cart and pendulums. The simulation program can draw the double inverted pendulum system and its SIRM fuzzy controller based on multi-objective particle swarm optimization. This action occurred every 0.01 s; so the Java animation would be created as shown in Fig. 7.

TABLE IV
MAINDIPCLASS USED IN JAVA CODE

```
Package javadoubleinvepend;
Public class MainDIPClass extends Applet implements Runnable {
Thread P1 = null;
boolean ThreadD;
public void init(){· · ·}
public void run(){· · ·}
public void start(){· · ·}
public void stop(){· · ·}
public void paint(){· · ·}
public void DoubleInvPend(){· · ·}
```

TABLE V
SIRMCLASS USED TO IMPLEMENT THE EQUATIONS IN SECTION II-A

```
Package javadoubleinvepend;
Public class SIRMClass {
    MainDIPClass Obj = new MainDIPClass();
    Public static double MainSIRMMeth(int index)
    {
        for (int i=1; i <7; i++)
            nxi[i] = MainDIPClass. Nx[i];
        for(int i=1; i <7; i++)
        {
            if(nxi[i] < −1)
            {
                NB[i] = 1;
                ZO[i] = 0;
                PB[i] = 0;
            }
            else if (nxi[i] <= 0)
            {
                NB[i] = −nxi[i];
                ZO[i] = nxi[i]+1;
                PB[i] = 0;
            }
            else if (nxi[i]<=1)
            {
                NB[i] = 0;
                ZO[i] = −nxi[i]+1;
                PB[i] =nxi[i];
            }
            else if (nxi[i] > 1)
            {
                NB[i] = 0;
                ZO[i] = 0;
                PB[i] = 1;
            }
        }
fi[index]=(NB[index]*f_NB[index]+ZO[index]*f_ZO[index]+
PB[index]*f_PB[index])/(NB[index]+ZO[index]+PB[index]);
        return fi[index];
    }
}
```

The variables of the DoubleInvPend() method are introduced in Table VIII. Fig. 7 shows the results of Java program simulation. For this animation, the design variables of point C in Section V are used. This figure and attached video file reveal when the pendulums are going to fall, the cart is moving to the same side until the pendulums go up. As it is seen from Fig. 7, the cart will reach the desired position at the end of simulation. These observations reinforce the obtained simulation results in Section V.

TABLE VI
DIDCLASS USED TO IMPLEMENT THE EQUATIONS IN SECTION II-B

```
Package javadoubleinvepend;
Public class DIDClass
{
Public static double DID(double N_x1,double N_x3, int i)
    {
        double p = Math.abs(N_x1);
        if (p <= 0.5)
        {
            DS1 = –2*p+1;
            DM1 = 2*p;
            DB1 = 0;
        }
        else if (p <= 1)
        {
            DS1 = 0;
            DM1 = –2*p+2;
            DB1 = 2*p − 1;
        }
        double q = Math.abs(N_x3);
        if(q <= 0.5)
        {
            DS3 = –2*q+1;
            DM3 = 2*q;
            DB3 = 0;
        }
        else if (q <= 1)
        {
            DS3 = 0;
            DM3 = –2*q + 2;
            DB3 = 2*q − 1;
        }
        delta_wD[1] = (w_DS1*DS1+w_DM1*DM1+
          w_DB1*DB1)/(DS1+DM1+DB1);
        delta_wD[2] = delta_wD[1]; delta_wD[3] =
          (w_DS3*DS3+w_DM3*DM3+w_DB3*DB3)/
          (DS3+DM3+DB3);
        delta_wD[4] = delta_wD[3];
        delta_wD[5] = (w_DS1_DS3*(DS1*DS3)+w_DS1_DM3*
          (DS1*DM3) +w_DM1_DS3* (DM1*DS3))/
          ((DS1*DS3)+(DS1*DM3)+(DS1*DB3)+(DM1*DS3)+
          (DM1*DM3) +(DM1*DB3)+(DB1*DS3)+
          (DB1*DM3)+(DB1*DB3));
        delta_wD[6] = delta_wD[5];
        wD = w[i]+B[i]*delta_wD[i];
        return wD;
        }
}
```

TABLE XII
METHOD DOUBLEINVPEND USED IN JAVA CODE

```
public void DoubleInvPend()
{
    a11 = m1+m2+M;
    a12 = m1*l1*Math.cos(x1);
    a13 = m2*l2*Math.cos(x3);
    a21 = a12;
    a22 = 4*m1*l1*l1/3;
    a31 = a13;
    a33 = 4*m2*l2*l2/3;
    N_x[1] = x1/(Math.PI/12);
    N_x[2] = x2/(Math.PI/1.8);
    N_x[3] = x3/(Math.PI/12);
    N_x[4] = x4/(Math.PI/1.8);
    N_x[5] = x5/(2.4);
    N_x[6] = x6;
    for(int i = 1 ; i¡7;i++)
        wD[i] = DIDClass.DID(N_x[1],N_x[3],i);
    for(int i =1 ; i¡7;i++)
        fi[i] = SIRMClass.MainSIRMMeth(i);
    for(int j=1; j¡7; j++)
    {
        f+ = wD[j] * fi[j];
    }
    u = f*Zf;
    b1 = u+m1*l1*x2*x2*Math.sin(x1)+m2*l2*x4*x4*Math.sin(x3);
    b2 = m1*l1*gravity*Math.sin(x1);
    b3 = m2*l2*gravity*Math.sin(x3);
    dx1=x2;
    dx2 = (b2-a21*((b1*a22*a33-a12*b2*a33-a13*b3*a22)/(a11*a22*a33-a12*a21*a33-a13*a31*a22)))/a22;
    dx3 = x4;
    dx4 = (b3-a31*((b1*a22*a33-a12*b2*a33-a13*b3*a22)/(a11*a22*a33-a12*a21*a33-a13*a31*a22)))/a33;
    dx5 = x6;
    dx6 = (b1*a22*a33-a12*b2*a33-a13*b3*a22)/(a11*a22*a33-a12*a21*a33-a13*a31*a22);

    x1+ = deltaT*dx1;
    x2+ = deltaT*dx2;
    x3+ = deltaT*dx3;
    x4+ = deltaT*dx4;
    x5+ = deltaT *x6;
    x6+ = deltaT*dx6;
    posx = (int)(x5*coef);
    posxl1 = (int) (l1*coef * Math.sin(x1));
    posxl1 = (int) (l1*coef * Math.cos(x1));
    posxl2 = (int) (l2*coef * Math.sin(x3));
    posyl2 = (int) (l2*coef * Math.cos(x3));
            }
```

TABLE VIII
THE USED VARIABLES IN THE JAVA CODE

Variables	Description
u	Control effort
pos$x$	The position of the cart
pos$x$l1	The x component of the position of the longer pendulum
pos$x$l2	The x component of the position of the shorter pendulum
pos$y$l1	The y component of the position of the longer pendulum
pos$y$l2	The y component of the position of the shorter pendulum
coef	The coefficient of pixels to appropriate and balance number for design

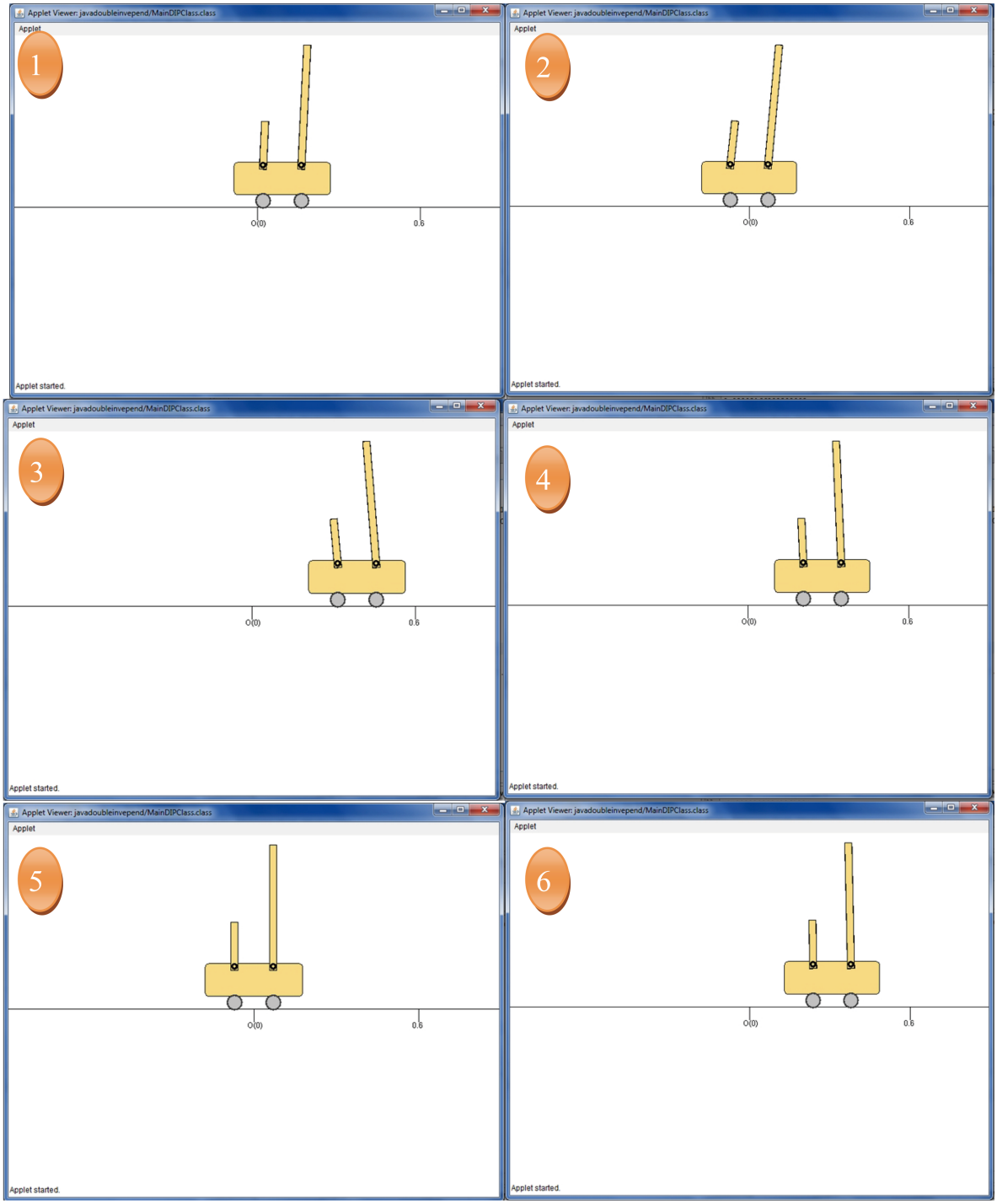

Fig. 7. Java simulation for the double inverted pendulum system.

IX. Conclusion

In this paper, an MOPSO algorithm was used, of which the leader selection method was based on density measures, the personal best positions were determined via Sigma method, the dynamic elimination technique was applied to prune the archive, and the turbulence operator was utilized to escape from the local minima. The proposed multi-objective algorithm was successfully used to obtain the Pareto frontiers of SIRMs dynamically connected fuzzy controllers for the double inverted pendulum system. The distance error of the cart and the summation errors of angles for the shorter and longer pendulums were considered as conflicting objective functions. The feasibility and efficiency of the proposed algorithm were shown in comparison with those obtained by two well-known multi-objective algorithms reported in the literature on this control problem. Furthermore, the modeling, control and stability simulation of the double inverted pendulum system were presented on the Net using Java Applets with educational goals.

REFERENCES

[1] Becerikli Y, Celik B K. Fuzzy control of inverted pendulum and concept of stability using Java application. *Mathematical and Computer Modelling*, 2007, **46**(1−2): 24−37

[2] Mahmoodabadi M J, Bagheri A, Arabani Mostaghim S, Bisheban M. Simulation of stability using Java application for Pareto design of controllers based on a new multi-objective particle swarm optimization. *Mathematical and Computer Modelling*, 2011, **54**(5−6): 1584−1607

[3] Liu Z, Zhang Y, Wang Y N. A type-2 fuzzy switching control system for biped robots. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 2007, **37**(6): 1202−1213

[4] Tseng C S, Chen B S. Robust fuzzy observer-based fuzzy control design for nonlinear discrete-time systems with persistent bounded disturbances. *IEEE Transactions on Fuzzy Systems*, 2009, **17**(3): 711−723

[5] Huang C H, Wang W J, Chiu C H. Design and implementation of fuzzy control on a two-wheel inverted pendulum. *IEEE Transactions on Industrial Electronics*, 2011, **58**(7): 2988−3001

[6] Zhu Y G. Fuzzy optimal control for multistage fuzzy systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 2011, **41**(4): 964−975

[7] Wang T, Zhang Y F, Qiu J B, Gao H J. Adaptive fuzzy backstepping control for a class of nonlinear systems with sampled and delayed measurements. *IEEE Transactions on Fuzzy Systems*, 2015, **23**(2): 302−312

[8] Yi J Q, Yubazaki N, Hirota K. A new fuzzy controller for stabilization of parallel-type double inverted pendulum system. *Fuzzy Sets and Systems*, 2002, **126**(1): 105−119

[9] Shook D A, Roschke P N, Lin P Y, Loh C H. GA-optimized fuzzy logic control of a large-scale building for seismic loads. *Engineering Structures*, 2008, **30**(2): 436−449

[10] Shayeghi H, Jalili A, Shayanfar H A. Multi-stage fuzzy load frequency control using PSO. *Energy Conversion and Management*, 2008, **49**(10): 2570−2580

[11] Bingül Z, Karahan O. A fuzzy logic controller tuned with PSO for 2 DOF robot trajectory control. *Expert Systems with Applications*, 2011, **38**(1): 1017−1031

[12] Ahmadi B, Ghamati M, Jamali A, Nariman-Zadeh N. Pareto robust reliability-based controller design for probabilistic uncertain systems using fuzzy rules. In: Proceedings of the 2011 IEEE International Symposium on Innovations in Intelligent Systems and Applications (INISTA). Istanbul: IEEE, 2011. 59−63

[13] Mahmoodabadi M J, Mottaghi M B S, Mahmodinejad A. Optimum design of fuzzy controllers for nonlinear systems using multi-objective particle swarm optimization. *Journal of Vibration and Control*, to be published

[14] Kennedy J, Eberhart R. Particle swarm optimization. In: Proceedings of the 1995 IEEE International Conference on Neural Networks. Perth, Australia: IEEE, 1995. 1942−1948

[15] Angeline P J. Using selection to improve particle swarm optimization. In: Proceedings of the 1998 IEEE World Congress on Computational Intelligence. The 1998 IEEE International Conference on Evolutionary Computation. Anchorage, AK: IEEE, 1998. 84−89

[16] Yoshida H, Kawata K, Fukuyama Y, Takayama S, Nakanishi Y. A particle swarm optimization for reactive power and voltage control considering voltage security assessment. *IEEE Transactions on Power Systems*, 2000, **15**(4): 1232−1239

[17] Chen X, Li Y M. A modified PSO structure resulting in high exploration ability with convergence guaranteed. *IEEE Transactions on Systems, Man, and Cybernetics, Part B, Cybernetics*, 2007, **37**(5): 1271−1289

[18] Chen Y P, Peng W C, Jian M C. Particle swarm optimization with recombination and dynamic linkage discovery. *IEEE Transactions on Systems, Man, and Cybernetics, Part B, Cybernetics*, 2007, **37**(6): 1460−1470

[19] Monson C K, Seppi K D. The Kalman swarm: a new approach to particle motion in swarm optimization. In: Proceedings of the 2004 Genetic and Evolutionary Computation Conference, Lecture Notes in Computer Science. Seattle, WA, USA: Springer, 2004. 140−150

[20] Janson S, Middendorf M. A hierarchical particle swarm optimizer and its adaptive variant. *IEEE Transactions on Systems, Man, and Cybernetics, Part B, Cybernetics*, 2005, **35**(6): 1272−1282

[21] Krohling R A, dos Santos Coelho L. Coevolutionary particle swarm optimization using Gaussian distribution for solving constrained optimization problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B, Cybernetics*, 2006, **36**(6): 1407−1416

[22] Van Den Bergh F, Engelbrecht A P. A new locally convergent particle swarm optimiser. In: Proceeding of the 2002 IEEE International Conference on Systems, Man, and Cybernetics. Yasmine Hammamet, Tunisia: IEEE, 2002.

[23] Zhang W J, Xie X F. DEPSO: hybrid particle swarm with differential evolution operator. In: Proceeding of the 2003 IEEE International Conference on Systems, Man, and Cybernetics. Washington, DC: IEEE, 2003. 3816−3821

[24] Hu X H, Eberhart R C. Multiobjective optimization using dynamic neighborhood particle swarm optimization. In: Proceedings of the 2002 Congress on Evolutionary Computation. Honolulu, HI: IEEE, 2002. 1677−1681

[25] Fieldsend J E, Singh S. A multi-objective algorithm based upon particle swarm optimisation, an efficient data structure and turbulence. In: Boceedings of the 2002 Workshop on Computational Intelligence, 2002. 34−44

[26] Mostaghim S, Teich J. Strategies for finding good local guides in multi-objective particle swarm optimization (MOPSO). In: Proceedings of the 2003 IEEE Swarm Intelligence Symposium. Indianapolis, IN, USA: IEEE, 2003. 26−33

[27] Parsopoulos K E, Tasoulis D K, Vrahatis M N. Multi-objective optimization using parallel vector evaluated particle swarm optimization. In: Proceedings of the 2004 IASTED International Conference on Artificial Intelligence and Applications. Austria, Calgary, Canada: ACTA Press, 2004. 823−828

[28] Yen G G, Leong W F. Dynamic multiple swarms in multiobjective particle swarm optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, 2009, **39**(4): 890−911

[29] Leong W F, Yen G G. PSO-based multiobjective optimization with dynamic population size and adaptive local archives. *IEEE Transactions on Systems, Man, and Cybernetics, Part B, Cybernetics*, 2008, **38**(5): 1270−1293

[30] Yubazaki N, Yi J Q, Otani M, Hirota K. SIRM's connected fuzzy inference model and its applications to first-order lag systems and second-order lag systems. In: Proceedings of the 1996 Asian Fuzzy Systems Symposium. Kenting: IEEE, 1996. 545−550

[31] Eberhart R, Dobbins R, Simpson P. *Computational Intelligence PC Tools*. San Diego, CA, USA: Academic Press Professional, Inc., 1996.

[32] Engelbrecht A P. *Computational Intelligence: An Introduction*. New York: John Wiley & Sons, 2002.

[33] Engelbrecht A P. *Fundamentals of Computational Swarm Intelligence*. New York: John Wiley & Sons, 2005.

[34] Eberhart R C, Shi Y H. Comparison between genetic algorithms and particle swarm optimization. In: Proceedings of the 7th International Conference. San Diego, California, USA: Springer, 1998. 611−616

[35] Eberhart R, Kennedy J. A new optimizer using particle swarm theory. In: Proceedings of the 6th International Symposium on Micro Machine and Human Science. Nagoya: IEEE, 1995. 39−43

[36] Ratnaweera A, Halgamuge S, Watson H C. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. *IEEE Transactions on Evolutionary Computation*, 2004, **8**(3): 240−255

[37] Tsai S J, Sun T Y, Liu C C, Hsieh S T, Wu W C, Chiu S Y. An improved multi-objective particle swarm optimizer for multi-objective problems. *Expert Systems with Applications*, 2010, **37**(8): 5872−5886

[38] Mostaghim S, Teich J. The role of ε-dominance in multi objective particle swarm optimization methods. In: Proceedings of the 2003 IEEE Congress on Evolutionary Computation. Canberra, ACT, Australia: IEEE, 2003. 1764−1771

[39] Wang Y J, Yang Y P. Particle swarm optimization with preference order ranking for multi-objective optimization. *Information Sciences*, 2009, **179**(12): 1944−1959

[40] Hernández-Díaz A G, Santana-Quintero L V, Coello Coello C A, Molina J, Caballero R. Improving the efficiency of ε-dominance based grids. *Information Sciences*, 2011, **181**(15): 3101−3129

[41] Mahmoodabadi M J, Arabani Mostaghim S, Bagheri A, Nariman-Zadeh N. Pareto optimal design of the decoupled sliding mode controller for an inverted pendulum system and its stability simulation via Java programming. *Mathematical and Computer Modelling*, 2013, **57**(5−6): 1070−1082

[42] Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGAII. *IEEE Transactions on Evolutionary Computation*, 2002, **6**(2): 182−197

[43] Gould H, Tobochnik J, Christian W. *An Introduction to Computer Simulation Methods: Applications to Physical Systems* (Third Edition). Reading, MA: Addison-Wesley Publishing, 2006.

[44] Deitel-Deitel H M, Deitel-Deitel P J. $Java^{TM}$ *How to Program* (Sixth Edition). New York: Prentice Hall Publishing, 2004.

[45] Hopson K C, Ingram S E. *Developing Professional Java Applets*. Indianapolis, IN, USA: Sams.net Publishing, 1996.

Mohammad Jawad Mahmoodabadi received his BS and MS degrees in mechanical engineering from Shahid Bahonar University of Kerman, Iran in 2005 and 2007, respectively. He received his Ph. D. degree in mechanical engineering from the University of Guilan, Rasht, Iran in 2012. He worked for 2 years in the Iranian Textile industries. During his research, he was a scholar visitor with Robotics and Mechatronics Group, University of Twente, Enchede, theNetherlands for 6 months. Now, he is an assistant professor of Mechanical Engineering at Sirjan University of Technology, Sirjan, Iran. His research interests include optimizational gorithms,non- linear and robust control, and computational methods. Corresponding author of this paper.

S. Arabani Mostaghim received her B. Sc. degree in software engineering from University of Guilan, Iran in 2012. She is now doing a M. Sc. program in the University of Guilan. Her research interests include optimization, cryptography, video tracking, and image processing.